

Contents

Abstract	5
Acknowledgements.....	6
Chapter 1: Introduction	7
1.1 Background	7
1.2 Problem Definition.....	7
1.3 Proposed Solution.....	7
1.4 Project Objectives	8
1.5 Strategy	8
Chapter 2: Literature Review	10
2.1 Programmable Logic Controllers.....	10
2.1.2 Components of an Industrial PLC.....	11
2.1.3 PLC Programming.....	11
2.1.4 Commercial PLCs.....	12
2.1.5 Commercial PLC Software.....	13
2.1.6 Free PLC Software	14
2.2 The IEC 61131 Standard PLC	14
2.2.1 Memory for application data storage	15
2.2.3 Programming Languages.....	16
2.2.4 IEC 61131-3 POUs	16
2.3 INTEGRATED DEVELOPMENT ENVIRONMENT	17
2.3.1 MICROCONTROLLER IDEs.....	18
2.3.2 Desktop Software IDEs	18
2.4 JavaFX Programming Language.....	18
2.4.1 JavaFX architecture	19
2.4.2 Running a JavaFX application.....	19
2.4.3 Java FXML.....	21
2.4.4 JavaFX Drag and Drop	22
2.5 Embedded Computer Systems.....	23
2.5.1 The PIC Microcontroller	24
2.5.2 The PIC18F2550.....	24
2.5.3 The PIC16F887 Microcontroller	33

2.5.4 MCP4921 DAC	34
2.6 MikroC Hardware Libraries for PIC	36
2.6.1 ADC Library.....	36
2.6.2 Flash Memory Library	37
2.6.3. Lcd Library.....	38
2.6.4 SPI Library	40
2.6.5 UART Library	42
2.6.6 USB Library.....	44
2.7 USB Communication	45
2.7.1 The HID USB Class	46
2.7.2 Interfaces	47
2.7.3 Device Descriptor Structure.....	47
Chapter 3: Desktop Application Design	48
3.1 Software Specification for JPLC Editor	48
3.1.1 Global Functionality	48
3.1.3 Compiler Functionality.....	49
3.1.4 Communication Functionality.....	50
3.2 UML Modelling.....	50
3.2.1 Use case Diagram.....	50
3.2.2 Package Definition	51
3.2.3 Software Functionality Sequence Specification.....	53
3.3 USB Communication Protocol design	57
3.3.1 Protocol Communication Requirements	57
3.3.2 Protocol implementation	58
3.3.2.2 Protocol Representation	59
3.3.2.3 Protocol Sessions	60
Chapter 4: Hardware Design and Testing	63
4.1 Microcontroller Design	63
4.1.1 Pin assignment	63
4.1.2 User program microcontroller Flow Cart.....	66
4.1.3 I/O Interface Microcontroller Flow chart.....	67
4.1.4 Program Memory Assignment	67
4.1.5 Data Memory Assignment	68

4.2 Opto-Isolator Design	68
4.2.1 Input Opto-isolator	68
4.2.2 Output Opto-isolator	70
Chapter 5: Results and analysis	71
5.1 JPLC Editor Design Implementation and Testing	71
5.1.1 Default Graphical User Interface	71
5.1.2 Creating a new Project	73
5.1.3 Creating a tag	80
5.1.4 Creating a Rung	82
5.1.5 Dragging data onto the Ladder Diagram	84
5.1.6 Saving and Opening a Project	90
5.1.7 Compiling the Program	95
5.1.8 Downloading the program	97
5.2 Processor board implementation and Testing	99
5.2.1 Run Mode Test simulation	99
5.2.2 Program Mode Test	100
5.3 Opto-isolator boards Implementation and Testing	105
5.3.1 Digital Input Circuit Simulation	105
5.3.2 Digital Output Circuit Simulation	106
Chapter 6: Conclusion	108
6.1 Objectives Met	108
6.2 Problem Statement	109
6.3 Recommendations	109
References	110
Appendix A: User program microcontroller code	111
Appendix B: I/O interface microcontroller code	117
Appendix C: Run Mode Test user program microcontroller code	121
Appendix D: JPLC Editor main package classes code	123
Appendix E: JPLC Editor user interaction layer classes code	124
Java Classes	137
Appendix F: JPLC Editor resource layer classes code	154
Appendix G: JPLC Editor Project layer classes code	157
Appendix H: JPLC Editor Communication layer classes code	175

Appendix I: JPLC Editor Compiler layer classes code	191
Appendix J: JPLC Editor Ladder Diagram layer classes code.....	298
Ladder Diagram Instruction FXML codes	398
Appendix K: Project name compilation files	414
Project name.ld file contents	414
Project name.asm file contents	414
Project name.pgm file contents.....	415
Project name.hex file contents	418

Abstract

Programmable Logic Controllers (PLCs) are the heart of most control processes in industries. PLCs read electrical signals from sensors and perform a control algorithm depending on the user program. PLCs are normally programmed in Ladder Logic using a desktop application and the program is transferred to the PLC via a serial cable.

PLCs are made from affordable electronic components but are sold at high prices because manufacturers value them based on the amount of work they do in industry. As a result, the PLCs are too expensive for most education institutions to purchase for student training purposes. This PLC Emulator system provides the PLC training student an affordable platform where they can practice their Ladder Logic.

A desktop application programmed using Java programming language has been developed using the free software Netbeans IDE. The program allows the user to develop and compile a Ladder program and download it to the PLC emulator. The physical Emulator was designed using two microcontrollers. One contains the PLC operating system and stores the user program. The other provides an interface to the inputs and outputs. Both microcontroller programs were developed using a licenced version of the MikroC IDE.

This Emulator provides a practical training platform for the student and will create blueprints for a product that has never been manufactured in Zimbabwe.

While the government is cutting down on foreign spending, all PLCs used in Zimbabwe are imported from outside the country using up foreign currency. This project can help reduce foreign spending fulfilling some of the requirements of the recently published Zim-Asset document.

Acknowledgements

The author would like to sincerely thank his supervisors Mr. Reginald Gonye and Mr Lovemore Gunda for all their undying support and patience throughout this project.

The author would also like to thank the chairperson of the Department, Mrs N.B Nleya for securing funding from the University for this project making its final transfer to the prototyping board possible.

The author would also like to thank all the lecturing staff of the Electronics Department for the past five years of guidance.

The author would especially like to thank his mother and family for their love and encouragement.

Thank you.

Chapter 1: Introduction

This chapter introduces the write up of the project taken by the author from the problem definition and proposed solution to the expected design process.

1.1 Background

Programmable Logic Controllers (PLCs) are the heart of most control processes in industries. PLCs read electrical signals from sensors and perform a control algorithm depending on the user program. PLCs are normally programmed in Ladder Logic using a desktop application and the program is transferred to the PLC via a serial cable.

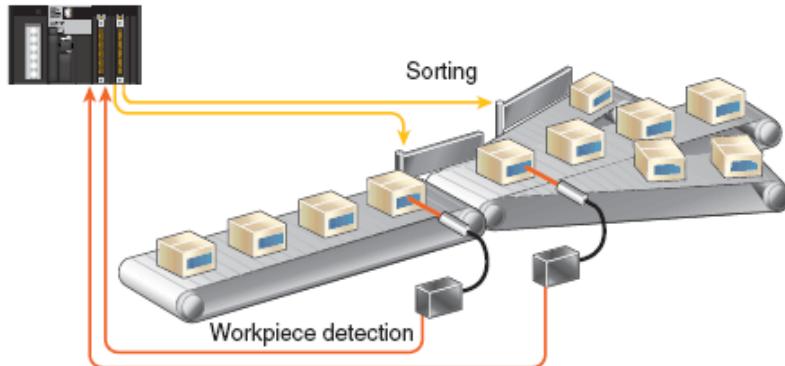


Figure 1.1 Siemens PLC Controlling a Process

1.2 Problem Definition

PLCs are made from affordable electronic components but are sold at high prices because manufacturers value them based on the amount of work they do in industry. As a result, the PLCs are too expensive for most education institutions to purchase for student training purposes.

1.3 Proposed Solution

This PLC Emulator will seek to provide the PLC training student an affordable platform where they can practice their ladder logic. A PLC Emulator will be designed from cheap microcontrollers from Microchip and affordable voltage isolators will be used to isolate the microcontroller from voltages as high as 24V that PLCs normally use. A compiler will also be designed that will allow the user to program the microcontroller using ladder logic.

This Emulator provides a practical training platform for the student and will create blueprints for a product that has never been manufactured in Zimbabwe.

1.4 Project Objectives

1. To design a software compiler that will allow the user to program a PIC microcontroller using ladder logic.
 - This software will allow the user to create a ladder diagram by means of a drag and drop interface.
 - This will also involve developing a data structure for the data types to be used for the PIC microcontroller to be chosen for the PLC Emulator.
 - It will also allow the user to verify the syntax of the ladder diagram where now a hexadecimal file will be generated for every compilation of the program. The software will also allow the user to download the program to the PLC emulator.
2. To design a microcontroller operating system that will allow the PLC emulator to enter run or program mode, as well as communicate with the compiler on a PC. This will involve developing a PIC microcontroller program which will indicate the status of the emulator on an LCD as well as handle communications with the software compiler.
3. To assemble the PLC emulator illustrated in Figure 1.2 consisting of:
 - A power supply that will supply 5V DC and 24V DC to the PLC emulator components
 - An interface to inputs and outputs with LED indicators indicating the status of each input or output. The inputs will also be isolated from the PIC microcontrollers via opto-isolators. Four digital inputs, four digital outputs, one analogue input and one analogue output will currently be proposed and these will be embedded on the PLC and not on any expansion cards. No expansion input or output card feature will be proposed now.
 - Two PIC microcontrollers communicating via serial bus sharing information as required by the user program and compiler on PC.
 - The emulator will first be assembled and simulated on Proteus then built on a breadboard.

1.5 Strategy

This project will seek to partially fulfil the following aspects of the IEC 61131 Standard PLC specification [20] as the project will only have less than a year to reach conclusion.

1. The power supply function.
2. The communication function.
3. The Programming, debugging, testing and documentation function.
4. A limited Interface function to sensors and actuators

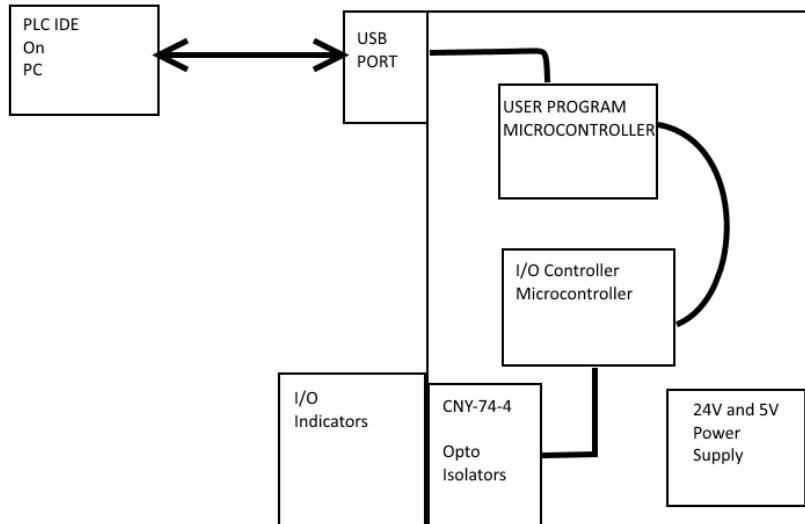


Figure 2.2 PLC Emulator System Block Diagram

Two PIC Microcontrollers will be used. One will handle communication with the compiler on PC as the communications controller and will also contain and run the user program. The second microcontroller will be responsible to reading and writing to the inputs and outputs then communicating this data to the main microcontroller.

A software compiler that will allow the user to program the PLC emulator will be the first thing to be developed. The compiler will allow the user to program in ladder logic and will be by means of a drag and drop interface allowing the user to create rungs for the program.

The conversion of ladder logic to a hexadecimal code will involve the following steps:

1. Scanning of the ladder node tree on the GUI using a Depth First Search which will result in the creation of an Instruction List (IL) [21].
2. Conversion of the instruction list into a medium level assembly language format which is native for PIC microcontrollers.
3. Conversion of the medium level Instruction List into hexadecimal format

Once the hexadecimal file is available, downloading it to the PLC will simply involve generating a binary stream from the hexadecimal data then transmitting it serially via a USB port to the PLC emulator.

At the PLC Emulator side, a PIC microcontroller program which will handle communication between the PLC Emulator system and the outside world will then be developed. This controller will also serve as the operating system for the emulator handling the power up as well as any user input allowing the emulator to enter program or run mode.

Chapter 2: Literature Review

This chapter looks at the research done by the author and the theories of the technologies used to come up with the design. Comparison of components and technologies is done here and parameters which led to the choice of components are clearly shown with clear reasons why the other technologies were rejected.

2.1 Programmable Logic Controllers

In simple terms a PLC is an Industrial microcontroller system (which may use conventional processors as well) adapted to the Industrial environment. They were first developed in the automobile industry to provide easily programmable controllers to replace hard-wired relays and timers [1].

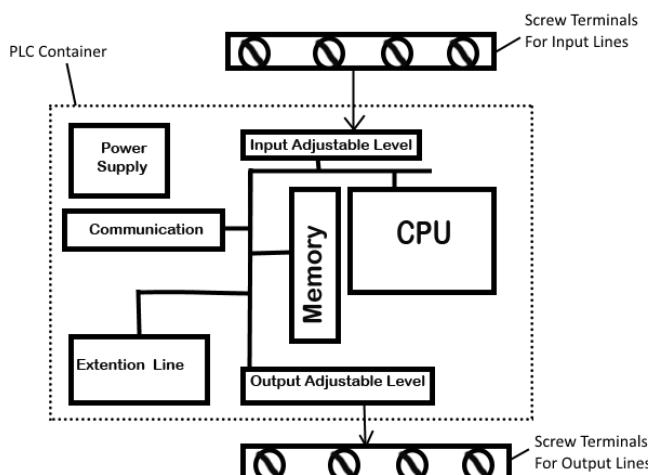


Figure 2.1 Internal Structure of a PLC

The input and output terminals have protection mechanisms to isolate the CPU from the potentially damaging industrial conditions. Some of these mechanisms are to reduce the input voltage the PLC receives to the 5V DC the processor can handle, and increase it from 5V DC at the outputs to the 24V/220V AC industrial machines use. An input adjustment interface reduces high voltages to 5V dc through opto-isolation.

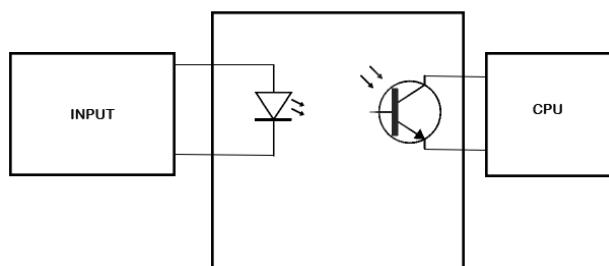


Figure 3.2 Opto-Isolation between the CPU and an input voltage

2.1.2 Components of an Industrial PLC

The core components of a PLC System include:

- Power supply
- Chassis
- Processor
- I/O modules

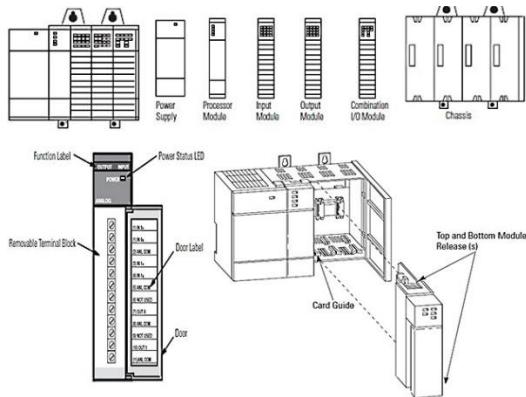


Figure 2.4 Components of a PLC

The Power supply is usually 220 V AC which is stepped down inside the power supply to the 24V DC the PLC uses. Some PLCs don't have power suppliers which step down the voltage and only accept 24V DC.

The chassis is the backbone of the PLC which connects the processor to its Input and Output modules. It runs a bus at the back to the rack and has sockets which connect each module to the chassis/ rack.

2.1.3 PLC Programming

PLCs can be programmed using structured text, function blocks or ladder logic. Ladder logic is usually the preferred way of programming PLCs. Ladder logic is a simple way of programming which makes the programmer focus on the task at hand only and very little time configuring the processor. Hence Ladder logic was the preferred programming language for this project.

Ladder logic is a symbolic representation of electric circuits in sequential execution. Execution continues down the rungs if continuity is established in each rung.

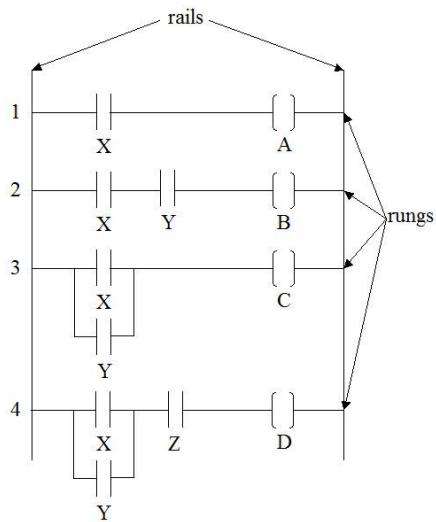


Figure 2.4 PLC Ladder Diagram components

The most common inputs/ sensors to a PLC are the Normally Open and Normally Closed contacts. A normally open contact will not conduct electricity until it is pressed down to allow current flow for example a push button switch. A normally closed contact will conduct current to an output relay until it is disconnected by a press or whatever input from the user.

A ladder program thus consists of conditions on the left which are either ON or OFF (logic 1 or logic 0) and outputs on the right or instructions inside function blocks, which will be executed if and only if the conditions produce continuity.

2.1.4 Commercial PLCs

A number of manufacturers of PLCs exist but Rockwell Automation is the world's largest company dedicated to Industrial Automation having offices in USA, China, Brazil, South Africa plus more. They have a wide range of PLCs suitable for a wide range of applications. Their simplest PLC is the Micrologix series of PLCs with the cheapest one being the Micrologix 1000 [22]. These PLCs have inbuilt digital and ports and still support expanding the Input/Output connections via and I/O bus. A Micrologix 1000 PLC would sell for a minimum of USD \$500 at Mine Elect Zimbabwe and the price would vary according to the ZIMRA rates.

The Micrologix 1000 series of PLCs have the following basic features:

- 1 KB combined user program memory.
- 10 to 32 Embedded digital I/O connections.
- Two analog current and two voltage inputs.
- One 8 pin RS232 communication port.
- 120/240V AC power.

This PLC was set as a benchmark target for this model of the PLC the author is to develop.

2.1.5 Commercial PLC Software

Commercially available PLC software like RS Logix 500 allow the programmer to create ladder logic programs, download the program to the PLC then see a live simulation of the running program while it is executing.

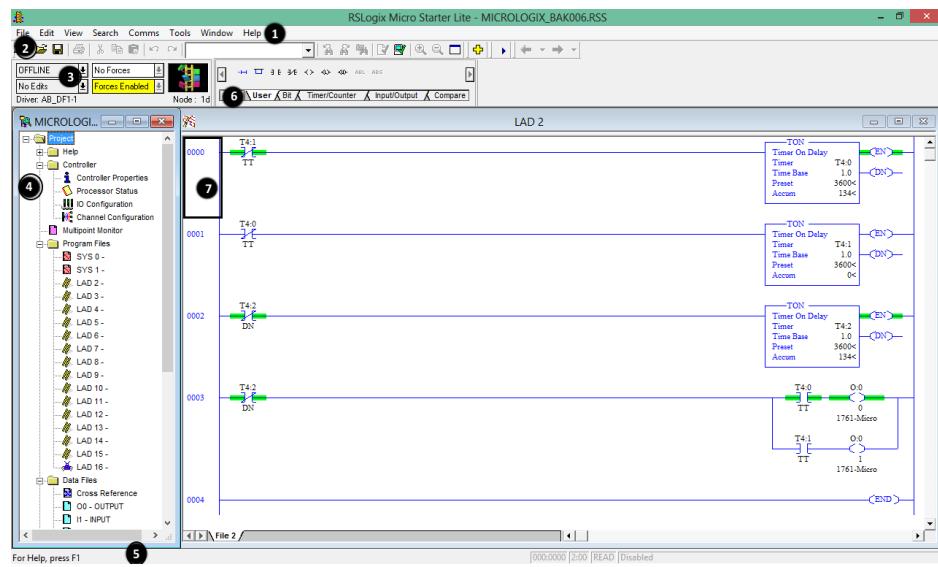


Figure 2.5 RSLogix 500 GUI Live Simulation of PLC program

The table below gives a brief description of the features available with commercial PLC software like RSLogix 500.

Table 1.1

Item	Name	Description
1	Menu bar	Menu selections.
2	Icon bar	The icon bar displays common functions. Hover your cursor over an icon to view a tool tip.
3	Online bar	Indicates the processor mode and whether online edits or forces are present, and the communications driver and node number.
4	Project tree	The project's folders and files. You can usually click an icon in this tree and then right-click to open a menu that applies to the item.
5	Status bar	Indicates status information and system prompts.
6	Instruction toolbar	Displays instruction mnemonics in tabbed categories. Click an instruction to insert it in your ladder program.
7	Ladder view	This is where you edit your ladder logic. You can view several program files at the same time.
not shown	Results pane	Displays the results of a search or a verification procedure at the bottom of the window. You can hide this pane or reposition it on your screen.

2.1.6 Free PLC Software

A number free PLC software exist and they all require you to have a PLC to program or build an expensive PLC from a number of components like OpenPLC project [3]. Another open source PLC software is LDmicro which allows you to program a PIC microcontroller or an arduino but its interface is nowhere near an industry standard software like RSLogix 500 like drag and drop interfaces.



Figure 2.6 LDmicro GUI

The goal of this project is to provide system which will give the user a good introduction to the working of software like RSLogix 500 while programming an affordable PIC microcontroller via processes one would use when using RSLogix.

2.2 The IEC 61131 Standard PLC

The IEC (International Electrotechnical Commission) is a worldwide organization for standardization to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. The IEC 61131-1 constitutes Part 1 of a series of standards on programmable controllers provided by this specification [20]. It applies to programmable controllers (PLC) and their associated peripherals such as programming and debugging tools (PDTs), human-machine interfaces (HMIs), and many more, which have as their intended use the control and command of machines and industrial processes.

This specification has set a standard for a PLC System with the following features:

- **The CPU function:** consists of the application programme storage, the data storage, the operating system, and the execution of the application programme functions.
- **The operating system function:** is responsible for the management of internal PLC-system interdependent functions (configuration control, diagnostics, memory management, application

programme execution management, communication with peripherals and with the interface functions to sensors and actuators.

- **Interface function to sensors and actuators converts:**
 - the input signals and/or data obtained from the machine/process to appropriate signal levels for processing.
 - the output signals and/or data from the signal processing function to appropriate signal levels to drive actuators and/or displays.
- **The Communication function:** provides data exchange with other systems (third-party devices) such as other PLC-systems, robot controllers, computers, etc.
- **Human-machine interface (HMI) function:** provides for interaction between the operator, the signal processing function and the machine/process.
- **Programming, debugging, testing and documentation function.** These functions provide for application programme generation and loading, monitoring, testing and debugging as well as for application programme documentation and archiving.
- **Power-supply functions:** provide for the conversion and isolation of the PLC-system power from the mains supply.

2.2.1 Memory for application data storage

2.2.1.1 Application program storage

The application program storage provides for memory locations to store a series of instructions whose periodic or event-driven execution determines the progression of the machine or the process.

2.2.1.2 Application data storage

The application data storage provides for memory locations to store I/O image table and data (for example, set values for timers, counters, alarm conditions, parameters and recipes for the machine or the process) required during the execution of the application program.

2.2.1.3 Memory type, memory capacity, memory utilization

Various types of memory are in use: read/write (RAM), read-only (ROM), programmable read-only (PROM), reprogrammable read-only (EPROM/UV-PROM, EEPROM). Memory retention at power failure is achieved by a proper selection of the memory type where applicable (for example, EPROM, EEPROM) or the use of memory back-up for volatile memories (for example, a battery).

2.2.2 Execution of the Application program

An application program may consist of a number of tasks. The execution of each task is accomplished sequentially, one programmable function at a time until the end of the task. The initiation of a task, periodically or upon the detection of an event (interrupt condition), is under the control of the operating system.

2.2.3 Programming Languages

For the programming of the application, there is a set of languages defined in IEC 61131-3.

Textual languages

1. Instruction list (IL) language. A textual programming language using instructions for representing the application program for a PLC-system.
2. Structured text (ST) language A textual programming language using assignment, sub-program control, selection and iteration statements to represent the application program for a PLC-system.

Graphical languages

1. Function block diagram (FBD) language. A graphical programming language using function block diagrams for representing the application program for a PLC-system.
2. Ladder diagram (LD) language. A graphical programming language using ladder diagrams for representing the application program for a PLC-system.
3. Sequential function chart (SFC). A graphical and textual notation for the use of steps and transitions to represent the structure of a program organization unit (program or function block) for a PLC-System. The transition conditions and the step action can be represented in a subset of the above-listed languages.

2.2.4 IEC 61131-3 POU

The IEC 61131-3 is a guideline for Programming PLCS which basically provides a benchmark that allows PLC manufacturers to assess how closely their programming system keeps to the standard.

Program Organization Units correspond to the program blocks, organizational blocks, sequence blocks and functional blocks of the conventional PLC. The 3 main types of POU's are:

1. Program – the main program including assignment to Inputs and outputs, global variables and access paths.
2. Function Block - block with input and output variables.
3. Function – Block with function value for extension of the basic PLC operation.

All POU's consists of:

1. Variables.
2. Instructions.

The variables can be calling interface variables like EN and ENO global or local variables.

2.2.4.1 The Function

The code part of a function is such that the same input(s) will always produce the same output and operate with no memory. The function can have one or more input values but always return one function value. Functions have two additional inputs and outputs. The Boolean EN (Enable input) and the Boolean ENO (Enable output).

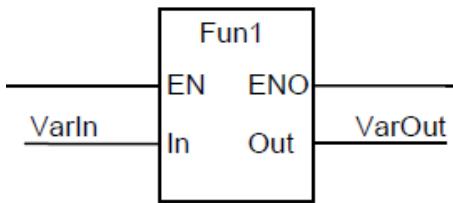


Figure 5 Example function

When EN = false, the code part of the function is not executed and ENO will be set to false. When EN = true, the code part is executed and ENO will initially be set to true before execution. ENO can later be set to true or false within the function. Examples of functions include:

- Data Type Conversion functions
- Numerical functions.
- Arithmetic Functions.
- Comparison Functions

2.2.4.2 The Function Block

A function block call results in the instantiation of a structure of variables which:

- Describe the function block interface like a data structure.
- Contain the actual states of the actual block.
- Represent a method for calling the function block.

A function block can have any number of input and output variables of none at all. Standard Function Blocks include:

- Counters.
- Timers.
- Communication function blocks.

2.2.4.3 The Program

The program can have these features:

- Declaration of directly represented variables to access the I/O address of the PLC.
- A program associated with a task within the configuration.

2.3 INTEGRATED DEVELOPMENT ENVIRONMENT

An integrated development environment (IDE) is a software program that provides comprehensive facilities for programmers for software programmers. It normally consists of a source code editor, build tools and a debugger and some provide code completion. The author will now take the reader through a walkthrough of the IDEs to be used in this project.

2.3.1 MICROCONTROLLER IDEs

The most common IDE's for developing programs for PIC are the MikroC IDE and the MPLABX IDE. MPLAB is a completely free software and offers free compilers so is the MikroC license for programs less than 2k. The only thing that sets them apart are the numerous built in libraries in MikroC for interfacing almost all known hardware to a PIC. It has libraries for LCD, RS 232 module, USB module, Keypad module reducing the amount of code required to initialize and use the hardware.

The mikroC PRO for PIC is a powerful, feature-rich development tool for PIC microcontrollers. It is designed to provide the programmer with the easiest possible solution to developing applications for embedded systems, without compromising performance or control. MikroC PRO for PIC provides a successful match featuring highly advanced IDE, ANSI compliant compiler, broad set of hardware libraries, comprehensive documentation, and plenty of ready-to-run examples. It is for this reason why MikroC will be used for developing PIC microcontroller projects for the Communications controller and Input/Output controller PICs.

2.3.2 Desktop Software IDEs

A number of IDEs are available for developing Desktop software using programming Languages like C++, C# and Java. Because the author is highly skilled in Java than either of the other languages, this write up will now only focus on Java IDEs.

Java is an object-oriented programming language meaning that it is a real world project based language that is easy to learn and very readable. More than two billion devices on a PC today run on Java and many operating systems readily support Java applications.

The most used Java IDE today is the Eclipse IDE but NetBeans in the author's opinion is a better option largely because it offers easy drag and drop features for creating AWT windows plus has a community of plug in developers just like Eclipse. But because the author is more familiar with NetBeans, it is more preferable.

2.4 JavaFX Programming Language

JavaFX is a Java library of a set of graphics and media packages that enables developers to design, create, test, debug, and deploy rich client applications that operate consistently across diverse platforms [1]. The JavaFX library was preferred over the usual JavaSwing Library for the following reasons:

1. **FXML and Scene Builder.** FXML is an XML-based declarative mark-up language for constructing a JavaFX application user interface. A designer can code in FXML or use JavaFX Scene Builder to interactively design the graphical user interface(GUI). Scene Builder generates FXML mark-up that can be ported to an IDE where a developer can add the business logic. This means that JavaFX separates code for the user interface from that of the actual Java code, making the Java code lesser and easier to read compared to swing which bundles the interface and java code in one java class.

2. **Swing interoperability.** Existing swing components can be integrated on any JavaFX component using the SwingNode class, making JavaFX effectively an upgraded swing.

On top of these, JavaFX has more features than the traditional Java Swing.

2.4.1 JavaFX architecture

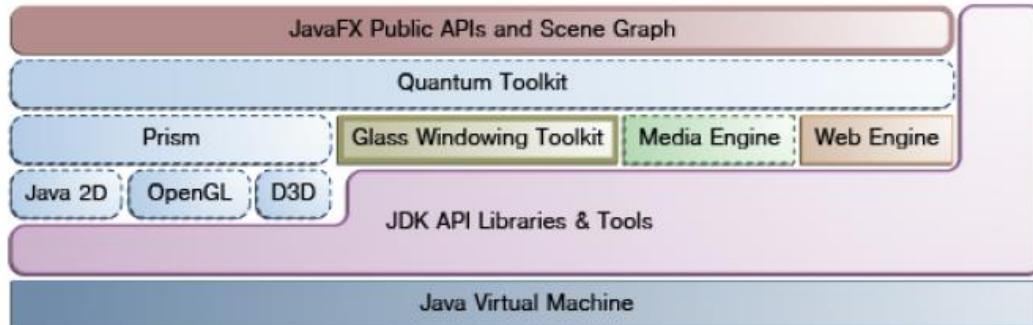


Figure 2.7 JavaFX Layers

The JavaFX scene graph, is the starting point for constructing a JavaFX application. A single element in a scene graph is called a node. Each node has an ID, style class, and bounding volume. With the exception of the root node of a scene graph, each node in a scene graph has a single parent and zero or more children. Each node is classified as either a *branch node* (meaning that it can have children), or a *leaf node* (meaning that it cannot have children) [2].

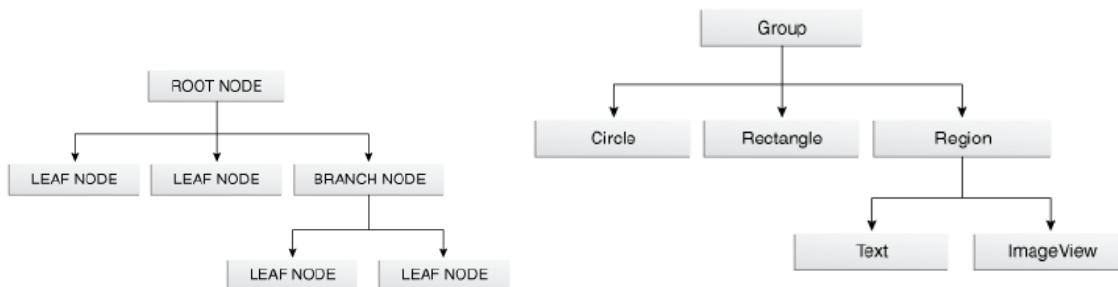


Figure 2.8 Node types and their specific classes

A Group object acts as the root node. The Circle and Rectangle objects are leaf nodes, because they do not (and cannot) have children. The Region object (which defines an area of the screen with children than can be styled using CSS) is a branch node that contains two more leaf nodes (Text and ImageView).

2.4.2 Running a JavaFX application

The Main class of a JavaFX application is an extension of the `javafx.application.Application` class. Its start method is overridden and receives a Stage object (a top-level GUI container) as its only parameter. A Stage is a window. The following code illustrates the working of a JavaFX Main class.

```
package scenegraphdemo;
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.stage.Stage;

public class Main extends Application {
@Override
public void start(Stage stage) {
Group root = new Group();
Scene scene = new Scene(root, 500, 500, Color.BLACK);
stage.setTitle("JavaFX Scene Graph Demo");
stage.setScene(scene);
stage.show();
}
public static void main(String[] args) {
launch(args);
}
}
```

The root node (in this case, an instance of the `javafx.scene.Group` class) is created and passed to the scene's constructor, along with the scene's width, height, and fill. The stage's title, scene, and visibility are all set. The main method invokes the `Application.launch()` method. This produces the following window, which can be populated with JavaFX components:



Figure 2.8 SceneGraphDemo class launch

2.4.3 Java FXML

FXML is an XML-based language that provides the structure for building a user interface separate from the application logic of the code [3]. From a Model View Controller (MVC) perspective, the FXML file that contains the description of the user interface is the view. The controller is a Java class, optionally implementing the Initializable class, which is declared as the controller for the FXML file. The model consists of domain objects, defined on the Java side, that you connect to the view through the controller.

A simple example of an FXML file is shown below:

```
<BorderPane>
<top>
<Label text="Page Title"/>
</top>
<center>
<Label text="Some data here"/>
</center>
</BorderPane>
```

This can be saved as an fxml file and loaded in the start method of the main class as follows:

```
@Override
public void start(Stage primaryStage) throws Exception {
    primaryStage.setTitle("border pane FXML Example");
    BorderPane myPane = (BorderPane) FXMLLoader.load(getClass().getResource
    ("test.fxml"));
    Scene myScene = new Scene(myPane);
    primaryStage.setScene(myScene);
    primaryStage.show();
}
```

Running such an application will yield the following result:



Figure 2.9 FXML class launch example

2.4.4 JavaFX Drag and Drop

A drag-and-drop operation is a data transfer between two objects: a gesture source and a gesture target. The gesture source and gesture target can be the following objects:

- Nodes
- Scenes

A drag-and-drop gesture happens as follows: The user clicks a mouse button on a gesture source, drags the mouse, and releases the mouse button on a gesture target. While dragging the data, the user gets visual feedback, which denotes locations that do not accept the data and, when over a target that accepts the data, gives a hint where to drop the data [4].

The data is transferred using a dragboard, which has the same interface as a system clipboard but is only used for the drag-and-drop data transfer.

Transfer modes define the type of transfer that happens between the gesture source and gesture target. Available transfer modes include COPY, MOVE, and LINK.

As a simple example, a drag and drop operation will take place between the following text objects:

```
final Text source = new Text(50, 100, "DRAG ME");
final Text target = new Text(300, 100, "DROP HERE");
```

The drag-and-drop gesture can only be started by calling the startDragAndDrop method inside the handler of the DRAG_DETECTED event on a gesture source. It is here that transfer modes supported by the gesture source are defined, and the data to be transferred is placed onto the dragboard.

```
source.setOnDragDetected(new EventHandler<MouseEvent>() {
    public void handle(MouseEvent event) {
        /* drag was detected, start a drag-and-drop gesture */
        /* allow any transfer mode */
        Dragboard db = source.startDragAndDrop(TransferMode.ANY);
        /* Put a string on a dragboard */
        ClipboardContent content = new ClipboardContent();
        content.putString(source.getText());
        db.setContent(content);
        event.consume();
    }
});
```

After the drag-and-drop gesture is started, any node or scene that the mouse is dragged over is a potential target to drop the data. You specify which object accepts the data by implementing the DRAG_OVER event handler.

```
target.setOnDragOver(new EventHandler<DragEvent>() {
    public void handle(DragEvent event) {
        /* data is dragged over the target */
        /* accept it only if it is not dragged from the same node
```

```
* and if it has a string data */
if (event.getGestureSource() != target &&
event.getDragboard().hasString()) {
/* allow for moving */
event.acceptTransferModes(TransferMode.MOVE);
}
event.consume();
});
});
```

During a drag-and-drop gesture, when the mouse pointer hovers over a target that fits the given drag-and-drop gesture, the target typically changes its appearance to provide a hint to the user where the data can be dropped.

When the drag gesture enters the boundaries of a potential gesture target, the target receives a DRAG_ENTERED event. When the drag gesture leaves the potential target's boundaries, the target receives a DRAG_EXITED event. You can use the DRAG_ENTERED and DRAG_EXITED event handlers to change the target's appearance in order to provide the visual feedback to the user.

```
target.setOnDragEntered(new EventHandler<DragEvent>() {
public void handle(DragEvent event) {
/* the drag-and-drop gesture entered the target */
/* show to the user that it is an actual gesture target */
if (event.getGestureSource() != target &&
event.getDragboard().hasString()) {
target.setFill(Color.GREEN);
}
event.consume();
}
});
};

target.setOnDragExited(new EventHandler<DragEvent>() {
public void handle(DragEvent event) {
/* mouse moved away, remove the graphical cues */
target.setFill(Color.BLACK);
event.consume();
}
});
```

2.5 Embedded Computer Systems

The area of electronics responsible for designing and implementing products containing microcontrollers is known as Embedded Computer Systems. Born out of the microprocessor which could only function properly on desktop computers, Microcontrollers are today found in cell phones, DVDs, washers and many other portable devices. What is the difference between a Microprocessor and a Microcontroller? By microprocessor it is meant the general purpose Microprocessors such as Intel's X86 family (8086, 80286, 80386, 80486, and the Pentium) or Motorola's 680X0 family (68000, 68010, 68020, 68030, 68040, etc.). These microprocessors contain no RAM, no ROM, and no I/O ports on the chip itself. For this reason, they are commonly referred to as general-purpose Microprocessors.

A system designer using a general-purpose microprocessor such as the Pentium or the 68040 must add RAM, ROM, I/O ports, and timers externally to make them functional. Although the addition of external RAM, ROM, and I/O ports makes these systems bulkier and much more expensive, they have the advantage of versatility such that the designer can decide on the amount of RAM, ROM and I/O ports needed to fit the task at hand. This is not the case with Microcontrollers.

A Microcontroller has a CPU (a microprocessor) in addition to a fixed amount of RAM, ROM, I/O ports, and a timer all on a single chip. In other words, the processor, the RAM, ROM, I/O ports and the timer are all embedded together on one chip; therefore, the designer may not need to add any external memory, I/O ports, or timer to it.

The microcontroller has spawned a smaller, cheaper type of processor called the PIC microcontroller. This device has a smaller number of interfacing lines, does not need the use of external memory, and contains a satisfactory amount of internal RAM for this project and it is for this reason a pic was preferred over an Intel microcontroller.

2.5.1 The PIC Microcontroller

The PIC Microcontroller was developed by the Microchip Company to provide a simplified version of the standard microcontroller, while at the same time utilizing special hardware techniques to produce a high level of performance. Microchip produces a range of devices to enable cost-effective solutions to a variety of problems. At the bottom of the ladder is the PIC12C5X series with 8-bit data and 12-bit instructions. The mid-range processors are the PIC16CXX series with 8-bit data and 14-bit instructions. The top end is occupied by the PIC18CXX devices with 8-bit data and 16-bit instructions. PIC 32XXX now exist with 32 bit data.

PIC16FXXX and PIC18FXXX family is a more modern family of PIC microcontrollers with a lot of suitable features being differentiated by pin counts and pin connections. The PIC16F887 with 40 pins has been found suitable for this project for the I/O controller, while the PIC18F2550 with just 28 pins and USB support was found suitable for the communications controller and user program pic.

2.5.2 The PIC18F2550

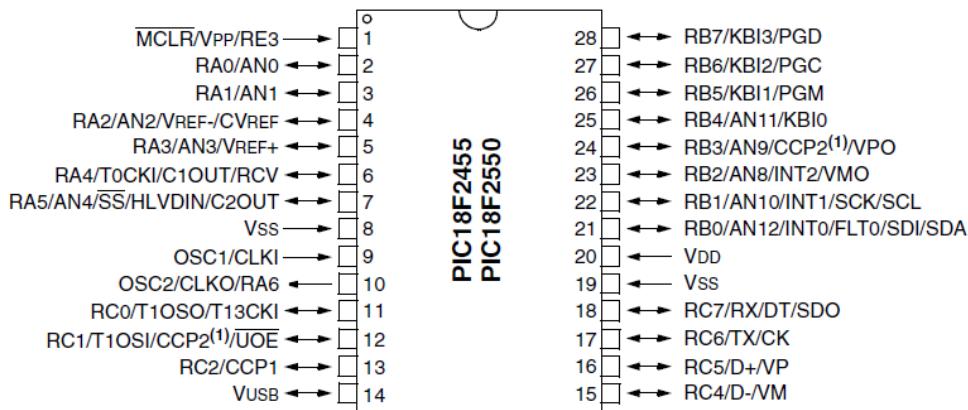


Figure 2.1.0 PIC18F2550 DIP IC

2.5.2.1 Features

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 Endpoints (16 bidirectional)
- 1-Kbyte Dual Access RAM for USB
- On-Chip USB Transceiver with On-Chip Voltage Regulator
- Interface for Off-Chip USB Transceiver
- Streaming Parallel Port (SPP) for USB streaming

Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 μ A typical
- Sleep mode currents down to 0.1 μ A typical
- Timer1 Oscillator: 1.1 μ A typical, 32 kHz, 2V
- Watchdog Timer: 2.1 μ A typical

Flexible Oscillator Structure:

- Four Crystal modes, including High Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal Oscillator Block:
 - 8 user-selectable frequencies, from 31 kHz to 8 MHz
 - User-tunable to compensate for frequency drift
- Secondary Oscillator using Timer1 @ 32 kHz
- Dual Oscillator options allow microcontroller and USB module to run at different clock speeds
- Fail-Safe Clock Monitor:
 - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High-Current Sink/Source: 25 mA/25 mA
- Three External Interrupts
- Four Timer modules (Timer0 to Timer3)
- Enhanced Capture/Compare/PWM (ECCP) module.
 - Multiple output modes
 - Selectable polarity
 - Programmable dead time
 - Auto-shutdown and auto-restart
- Enhanced USART module:
 - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI (all 4 modes) and I²C™ Master and Slave modes.

- 10-bit, up to 13-channel Analog-to-Digital Converter module (A/D) with Programmable Acquisition Time.
- Dual Analog Comparators with Input Multiplexing

Special Microcontroller Features:

- C Compiler Optimized Architecture with optional Extended Instruction Set
- 100,000 Erase/Write Cycle Enhanced Flash Program Memory typical
- 1,000,000 Erase/Write Cycle Data EEPROM Memory typical
- Flash/Data EEPROM Retention: > 40 years
- **Self-Programmable under Software Control**
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
 - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Wide Operating Voltage Range (2.0V to 5.5V)

Memory Features:

- 32Kbytes Flash program memory
- 2Kbytes of RAM
- 256 bytes EEPROM
- 10 Bit ADC

2.5.2.2 Program Memory Organization

The Flash program memory is readable, writable and erasable, during normal operation over the entire VDD range.

PIC18 devices have two interrupt vectors. The Reset vector address is at 0000h and the interrupt vector addresses are at 0008h and 0018h. The code memory space extends from 000000h to 007FFFh (32 Kbytes) in four 8-Kbyte blocks [6].

Addresses, 000000h through 0007FFh, however, define a “Boot Block” region that is treated separately from Block 0. All of these blocks define code protection boundaries within the code memory space.

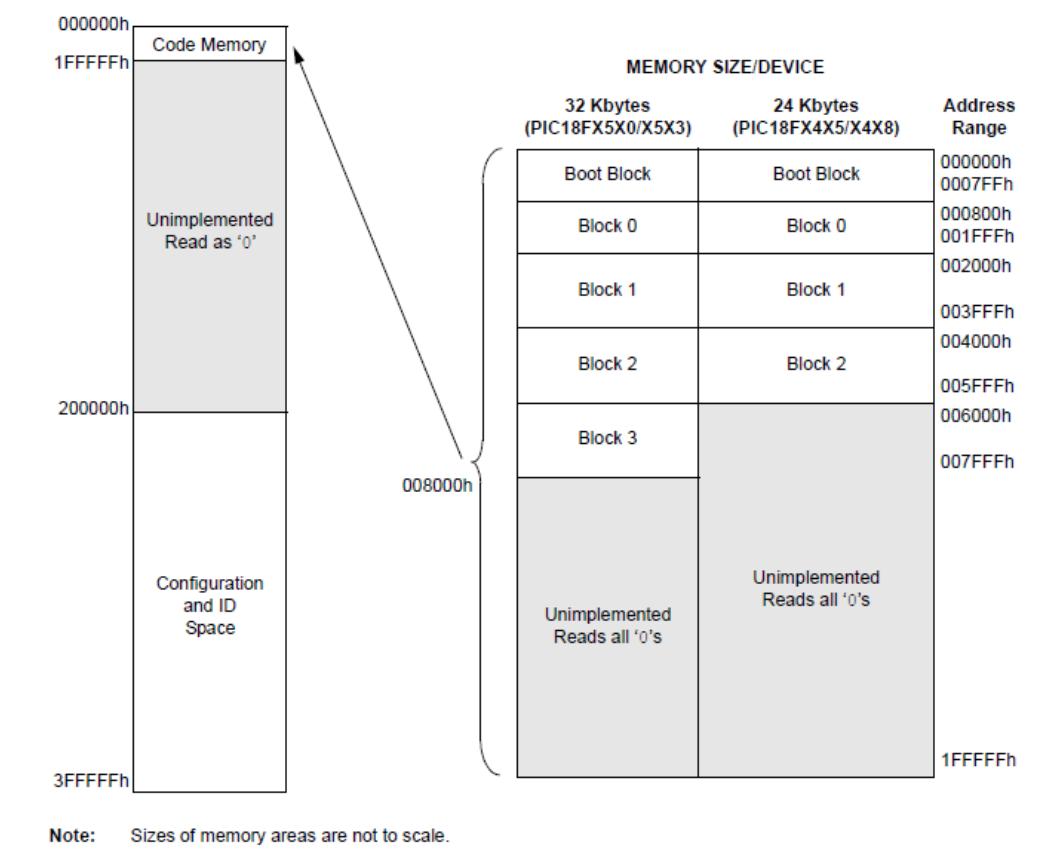


Figure 2.1.1 PIC 184550 and 2550 Program Memory Maps

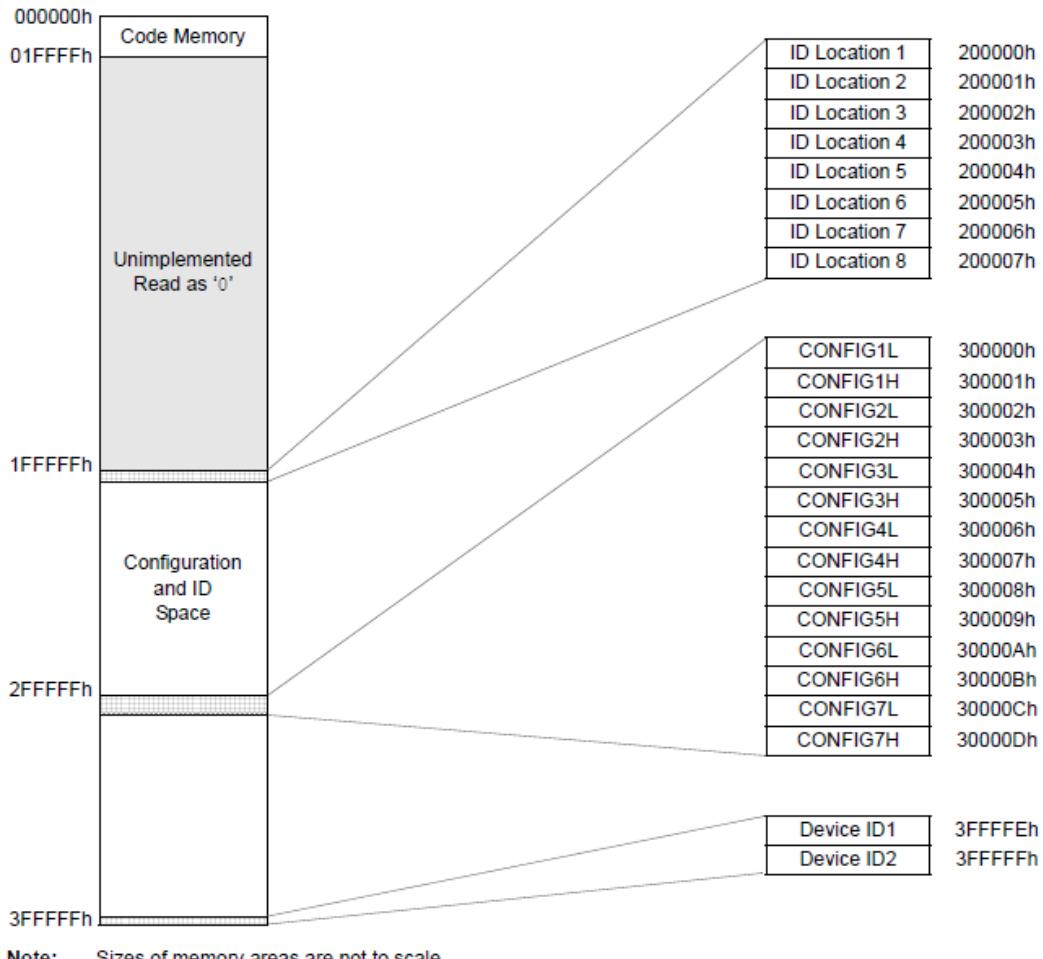


Figure 2.1.2 Pic18F2550 Program Memory Map

Memory in the address space, 0000000h to 3FFFFFFh, is addressed via the Table Pointer register, which is comprised of three pointer registers:

- TBLPTRU at RAM address 0FF8h
- TBLPTRH at RAM address 0FF7h
- TBLPTRL at RAM address 0FF6h

2.5.2.2.1 Instructions In Program Memory

The program memory is addressed in bytes. Instructions are stored as two bytes or four bytes in program memory. The Least Significant Byte of an instruction word is always stored in a program memory location with an even address (LSb = 0).

Program Memory Byte Locations →		Word Address ↓
	LSB = 1	LSB = 0
Instruction 1:	MOVLW 055h	000000h
Instruction 2:	GOTO 0006h	000002h
Instruction 3:	MOVFF 123h, 456h	000004h
	0Fh	000006h
	Efh	000008h
	F0h	00000Ah
	C1h	00000Ch
	F4h	00000Eh
		000010h
		000012h
		000014h

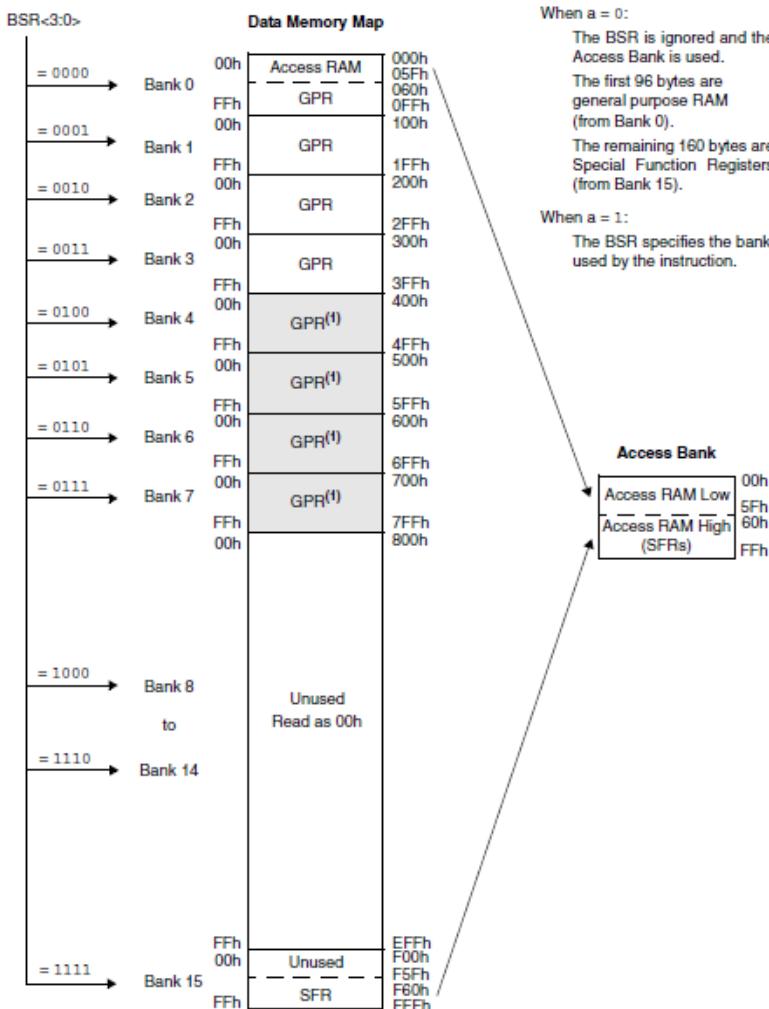
Figure 2.1.3 example of how instruction words are stored in the program memory.

2.5.2.3 Data Memory Organization

Each register in the data memory has a 12-bit address addressing up to 4096 bytes of data memory. The memory space is divided into 16 banks that contain 256 bytes each.

Banks 4 through 7 of the data memory are actually mapped to special dual port RAM. When the USB module is disabled, the GPRs in these banks are used like any other GPR in the data memory space. When the USB module is enabled, the memory in these banks is allocated as buffer RAM for USB operation.

Depending on the instruction, each location can be addressed directly by its full 12-bit address, or an 8-bit low-order address and a 4-bit Bank Pointer. Most instructions in the PIC18 instruction set make use of the Bank Pointer, known as the Bank Select Register (BSR). This SFR holds the 4 Most Significant bits of a location's address; the instruction itself includes the eight Least Significant bits. Only the four lower bits of the BSR are implemented (BSR3:BSR0). The BSR can be loaded directly by using the MOVLB instruction.



Note 1: These banks also serve as RAM buffer for USB operation. See **Section 5.3.1 "USB RAM"** for more information.

Figure 2.1.4 Pic 18F2550 RAM Organization

2.5.2.4 PIC18F2550 Instruction Set

Each single-word instruction is a 16-bit word divided into an opcode, which specifies the instruction type and one or more operands, which further specify the operation of the instruction. The instruction set is highly orthogonal and is grouped into four basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal** operations
- **Control** operations

Key:

1. d Destination select bit
 - a. d = 0: store result in WREG

- b. d = 1: store result in file register f
- 2. RAM access bit
 - a. a = 0: RAM location in Access RAM (BSR register is ignored)
 - b. a = 1: RAM bank is specified by BSR register
- 3. f = 8-bit register file address (00h to FFh) or 2-bit FSR designator (0h to 3h).
- 4. fs = 12-bit register file address (000h to FFFh). This is the source address.
- 5. fd = 12-bit register file address (000h to FFFh). This is the destination address.
- 6. k= Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value).
- 7. s=Fast Call/Return mode select bit
 - a. s = 0: do not update into/from shadow registers
 - b. s = 1: certain registers loaded into/from shadow registers (Fast mode)

Table 2

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes	
			MSb		Lsb				
BYTE-ORIENTED OPERATIONS									
ADDWF f, d, a	Add WREG and f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N	1, 2	
ADDWFC f, d, a	Add WREG and Carry bit to f	1	0010	00da	ffff	ffff	C, DC, Z, OV, N	1, 2	
ANDWF f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N	1, 2	
CLRF f, a	Clear f	1	0110	101a	ffff	ffff	Z	2	
COMF f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N	1, 2	
CPFSEQ f, a	Compare f with WREG, skip =	1 (2 or 3)	0110	001a	ffff	ffff	None	4	
CPFSGT f, a	Compare f with WREG, skip >	1 (2 or 3)	0110	010a	ffff	ffff	None	4	
CPFSLT f, a	Compare f with WREG, skip <	1 (2 or 3)	0110	000a	ffff	ffff	None	1, 2	
DECF f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4	
DECFSZ f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010	11da	ffff	ffff	None	1, 2, 3, 4	
DCFSNZ f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None	1, 2	
INCF f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4	
INCFSZ f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None	4	
INFSNZ f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100	10da	ffff	ffff	None	1, 2	
IORWF f, d, a	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N	1, 2	
MOVF f, d, a	Move f	1	0101	00da	ffff	ffff	Z, N	1	
MOVFF f _s , f _d	Move f _s (source) to 1st word f _d (destination) 2nd word	2	1100	ffff	ffff	ffff	None		
			1111	ffff	ffff	ffff			
MOVWF f, a	Move WREG to f	1	0110	111a	ffff	ffff	None		
MULWF f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None	1, 2	
NEGF f, a	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N		
RLCF f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N	1, 2	
RLNCF f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N		
RRCF f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N		
RRNCF f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N		
SETF f, a	Set f	1	0110	100a	ffff	ffff	None	1, 2	
SUBFWB f, d, a	Subtract f from WREG with borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N		
SUBWF f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	1, 2	
SUBWFB f, d, a	Subtract WREG from f with borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N		
SWAPF f, d, a	Swap nibbles in f	1	0011	10da	ffff	ffff	None	4	
TSTFSZ f, a	Test f, skip if 0	1 (2 or 3)	0110	011a	ffff	ffff	None	1, 2	
XORWF f, d, a	Exclusive OR WREG with f	1	0001	10da	ffff	ffff	Z, N		

Table 2.2 PIC18F2550 Byte-Oriented operations

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes	
			MSb	LSb					
BIT-ORIENTED OPERATIONS									
BCF f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2	
BSF f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2	
BTFSC f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4	
BTFSS f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4	
BTG f, d, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2	
CONTROL OPERATIONS									
BC n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None		
BN n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None		
BNC n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None		
BNN n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None		
BNOV n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None		
BNZ n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None		
BOV n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None		
BRA n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None		
BZ n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None		
CALL n, s	Call subroutine 1st word 2nd word	2	1110	110s	kkkk	kkkk	None		
			1111	kkkk	kkkk	kkkk			
CLRWDT —	Clear Watchdog Timer	1	0000	0000	0000	0100	TO, PD		
DAW —	Decimal Adjust WREG	1	0000	0000	0000	0111	C		
GOTO n	Go to address 1st word 2nd word	2	1110	1111	kkkk	kkkk	None		
			1111	kkkk	kkkk	kkkk			
NOP —	No Operation	1	0000	0000	0000	0000	None		
NOP —	No Operation	1	1111	xxxx	xxxx	xxxx	None	4	
POP —	Pop top of return stack (TOS)	1	0000	0000	0000	0110	None		
PUSH —	Push top of return stack (TOS)	1	0000	0000	0000	0101	None		
RCALL n	Relative Call	2	1101	1nnn	nnnn	nnnn	None		
RESET	Software device Reset	1	0000	0000	1111	1111	All		
RETFIE s	Return from interrupt enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL		
RETLW k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None		
RETURN s	Return from Subroutine	2	0000	0000	0001	001s	None		
SLEEP —	Go into Standby mode	1	0000	0000	0000	0011	TO, PD		

Table 2.3 PIC18F2550 Bit and Control Oriented Operations

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
			MSb			LSb		
LITERAL OPERATIONS								
ADDLW k	Add literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW k	AND literal with WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW k	Inclusive OR literal with WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR f, k	Move literal (12-bit) 2nd word to FSR(f) 1st word	2	1110	1110	00ff	kkkk	None	
MOVLB k	Move literal to BSR<3:0>	1	0000	0001	0000	kkkk	None	
MOVLW k	Move literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW k	Multiply literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW k	Subtract WREG from literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW k	Exclusive OR literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	
DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS								
TBLRD*	Table Read	2	0000	0000	0000	1000	None	
TBLRD*+	Table Read with post-increment		0000	0000	0000	1001	None	
TBLRD*-	Table Read with post-decrement		0000	0000	0000	1010	None	
TBLRD+*	Table Read with pre-increment		0000	0000	0000	1011	None	
TBLWT*	Table Write	2	0000	0000	0000	1100	None	
TBLWT*+	Table Write with post-increment		0000	0000	0000	1101	None	
TBLWT*-	Table Write with post-decrement		0000	0000	0000	1110	None	
TBLWT+*	Table Write with pre-increment		0000	0000	0000	1111	None	

Table 2.4 PIC18F2550 Literal and Program memory operations

2.5.3 The PIC16F887 Microcontroller

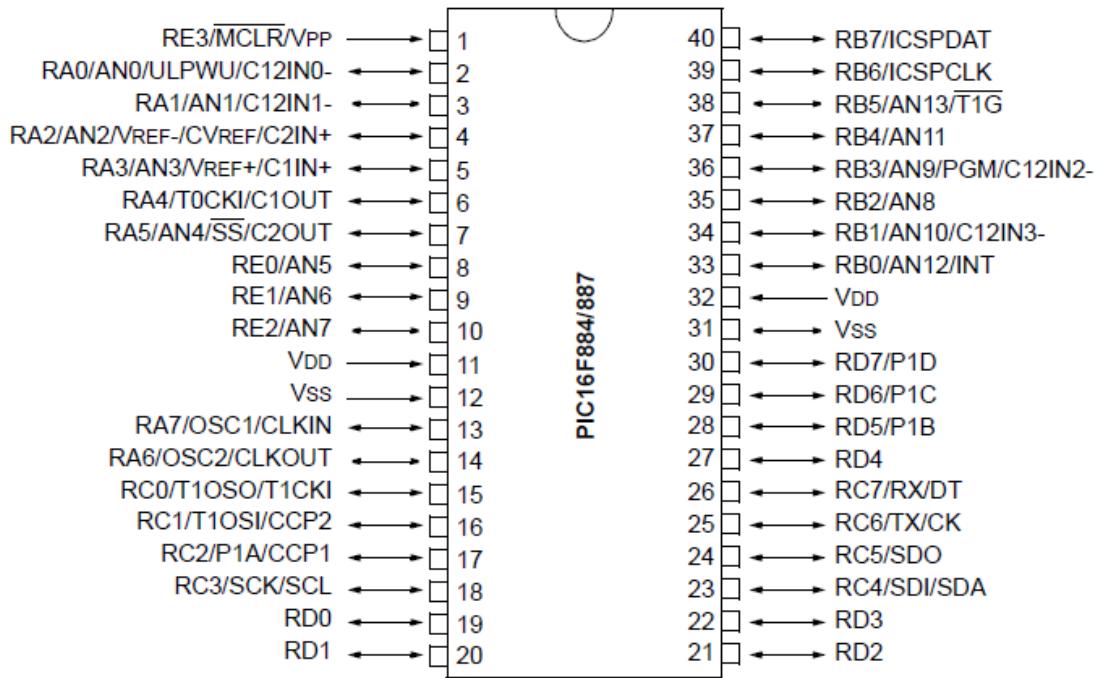


Figure 2.1.5 Pic16F887 pin diagram

Features

- Only 35 instructions to learn all single-cycle instructions except branches
 - Operating frequency 0-20 MHz
 - Precision internal oscillator
- Factory calibrated
- Software selectable frequency range of 8MHz to 31KHz
- Power supply voltage 2.0-5.5V
 - Consumption: 220uA (2.0V, 4MHz), 11uA (2.0 V, 32 KHz) 50nA (stand-by mode)
- Power-Saving Sleep Mode
- Brown-out Reset (BOR) with software control option
- 35 input/output pins
- High current source/sink for direct LED drive
- software and individually programmable *pull-up* resistor
 - Interrupt-on-Change pin
- 8K ROM memory in FLASH technology
 - Chip can be reprogrammed up to 100.000 times
- *In-Circuit Serial Programming* Option
 - Chip can be programmed even embedded in the target device
- 256 bytes EEPROM memory
 - Data can be written more than 1.000.000 times
- 368 bytes RAM memory
- Watch-dog timer
- Enhanced USART module
 - Supports RS-485, RS-232 and LIN2.0
 - Auto-Baud Detect
- Master Synchronous Serial Port (MSSP) Module supporting 3-wire SPI (all 4 modes) and I2C™ Master and Slave Modes with I2C Address Mask

2.5.4 MCP4921 DAC

The Microchip Technology Inc. MCP4921 is a 2.7 – 5.5V, low-power, low DNL, 12-Bit Digital-to-Analog Converter.

Features:

- 12-Bit Resolution
- ± 0.2 LSB DNL (typ)
- ± 2 LSB INL (typ)
- Single or Dual Channel
- Rail-to-Rail Output
- SPI™ Interface with 20 MHz Clock Support
- Simultaneous Latching of the Dual DACs w/LDAC
- Fast Settling Time of 4.5 μ s
- Selectable Unity or 2x Gain Output

- 450 kHz Multiplier Mode
- External VREF Input
- 2.7V to 5.5V Single-Supply Operation
- Extended Temperature Range: -40°C to +125°C

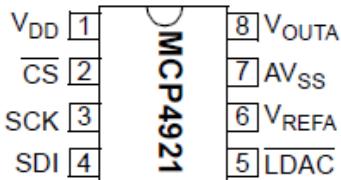


Figure 2.1.6 MCP4921 8 pin DIP package

2.5.4.1 Pin descriptions

- **V_{DD}** Positive Power Supply Input (2.7V to 5.5V)
- **CS** Chip Select Input
- **SCK** Serial Clock Input
- **SDI** Serial Data Input
- **LDAC** Syncronization input used to transfer DAC settings from serial latches to the output latches.
- **V_{REF}A** DACA Voltage Input (AV_{SS} to V_{DD})
- **AV_{SS}** Analog ground
- **V_{OUT}A** DACA Output

2.5.4.2 Serial Interface

The MCP492X family is designed to interface directly with the Serial Peripheral Interface (SPI) port, available on many microcontrollers, and supports Mode 0,0 and Mode 1,1. Commands and data are sent to the device via the SDI pin, with data being clocked-in on the rising edge of SCK.

Write Command Register

Upper Half:							
W-x	W-x	W-x	W-0	W-x	W-x	W-x	W-x
Ā/B	BUF	GA	SHDN	D11	D10	D9	D8
bit 15							bit 8

Lower Half:							
W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
D7	D6	D5	D4	D3	D2	D1	D0
bit 7							bit 0

Figure 2.1.7 Write Command Register

- bit 15 **A/B**: DACA or DACB Select bit
 - 1 = Write to DACB
 - 0 = Write to DACA
- bit 14 **BUF**: VREF Input Buffer Control bit
 - 1 = Buffered
 - 0 = Unbuffered

- bit 13 **GA**: Output Gain Select bit
 - 1 = 1x ($V_{OUT} = V_{REF} * D/4096$)
 - 0 = 2x ($V_{OUT} = 2 * V_{REF} * D/4096$)
- bit 12 **SHDN**: Output Power Down Control bit
 - 1 = Output Power Down Control bit
 - 0 = Output buffer disabled, Output is high impedance
- bit 11-0 **D11:D0**: DAC Data bits 12 bit number “D” which sets the output value. Contains a value between 0 and 4095.

Write Command

The write command is initiated by driving the CS pin low, followed by clocking the four configuration bits and the 12 data bits into the SDI pin on the rising edge of SCK. The CS pin is then raised, causing the data to be latched into the selected DAC's input registers.

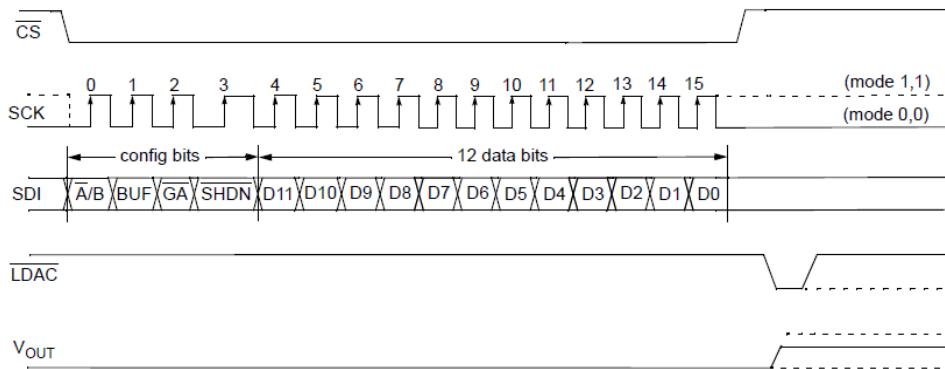


Figure 6 Write sequence

All writes to the MCP492X are 16-bit words. Any clocks past 16 will be ignored. The most significant four bits are configuration bits. The remaining 12 bits are data bits.

2.6 MikroC Hardware Libraries for PIC

2.6.1 ADC Library

Library Routines:

Prototype	void ADC_Init();
Returns	Nothing.
Description	This routine initializes PIC's internal ADC module to work with RC clock. Clock determines the time period necessary for performing AD conversion (min 12TAD).
Requires	MCU with built-in ADC module.
Example	ADC_Init(); // Initialize ADC module with default settings

Table 2.5 ADC_Init Function description

Prototype	<code>unsigned ADC_Read(unsigned short channel);</code>
Returns	10 or 12-bit unsigned value read from the specified channel (MCU dependent).
Description	<p>Initializes PIC's internal ADC module to work with RC clock. Clock determines the time period necessary for performing AD conversion (min 12TAD).</p> <p>Parameter channel represents the channel from which the analog value is to be acquired. Refer to the appropriate datasheet for channel-to-pin mapping.</p> <p>Note : This function doesn't work with the external voltage reference source, only with the internal voltage reference.</p>
Requires	The MCU with built-in ADC module. Before using the function, be sure to configure the appropriate TRISx bits to designate pins as inputs.
Example	<pre>unsigned tmp; ... tmp = ADC_Read(2); // Read analog value from channel 2</pre>

Table 2.6 ADC_Read Function description

2.6.2 Flash Memory Library

This library provides routines for accessing microcontroller Flash memory.

Library Routines:

Prototype	<code>//for PIC16</code> <code>unsigned FLASH_Read(unsigned address);</code> <code>//for PIC18</code> <code>unsigned short FLASH_Read(long address);</code>
Returns	Returns data byte from Flash memory.
Description	Reads data from the specified address in Flash memory.
Requires	Nothing.
Example	<code>//for PIC18</code> <code>unsigned short tmp;</code> <code>...</code> <code>tmp = FLASH_Read(0x0D00);</code> <code>...</code>

Table 2.7 Flash_Read Function description

Prototype	<code>void FLASH_Read_N_Bytes(long address, char* data_, unsigned int N);</code>
Returns	Nothing.
Description	Reads N data from the specified address in Flash memory to variable pointed by data
Requires	Nothing.
Example	<code>FLASH_Read_N(0x0D00,data_buffer,sizeof(data_buffer));</code>

Table 2.8 Flash_Read_N_Bytes

Prototype	<pre>//for PIC16 void FLASH_Write(unsigned address, unsigned int* data); //for PIC18 void FLASH_Write_8(long address, char* data); void FLASH_Write_16(long address, char* data); void FLASH_Write_32(long address, char* data); void FLASH_Write_64(long address, char* data);</pre>
Returns	Nothing.
Description	<p>Writes block of data to Flash memory. Block size is MCU dependent.</p> <p>P16: This function may erase memory segment before writing block of data to it (MCU dependent). Furthermore, memory segment which will be erased may be greater than the size of the data block that will be written (MCU dependent). Therefore it is recommended to write as many bytes as you erase. FLASH_Write writes 4 flash memory locations in a row, so it needs to be called as many times as it is necessary to meet the size of the data block that will be written.</p> <p>P18: This function does not perform erase prior to write.</p>
Requires	Flash memory that will be written may have to be erased before this function is called (MCU dependent). Refer to MCU datasheet for details.
Example	<pre>Write consecutive values in 64 consecutive locations, starting from 0x0D00: unsigned short toWrite[64]; ... // initialize array: for (i = 0; i < 64; i++) toWrite[i] = i; // write contents of the array to the address 0x0D00: FLASH_Write_64(0x0D00, toWrite);</pre>

Table 2.9 Flash_Write Function description

Prototype	<pre>//for PIC16 void FLASH_Erase(unsigned address); //for PIC18 void FLASH_Erase_64(long address); void FLASH_Erase_1024(long address);</pre>
Returns	Nothing.
Description	Erases memory block starting from a given address. For P16 family is implemented only for those MCU's whose flash memory does not support erase-and-write operations (refer to datasheet for details).
Requires	Nothing.
Example	Erase 64 byte memory memory block, starting from address 0x0D00: <code>FLASH_Erase_64(0x0D00);</code>

Table 2.1.0 Flash_Erase Function description

2.6.3. Lcd Library

The mikroC PRO for PIC provides a library for communication with Lcds (with HD44780 compliant controllers) through the 4-bit interface.

Library Routines:

Prototype	void Lcd_Init();
Returns	Nothing.
Description	Initializes Lcd module.
Requires	<p>Global variables:</p> <p>LCD_D7: Data bit 7 LCD_D6: Data bit 6 LCD_D5: Data bit 5 LCD_D4: Data bit 4 LCD_RS: Register Select (data/instruction) signal pin LCD_EN: Enable signal pin LCD_D7_Direction: Direction of the Data 7 pin LCD_D6_Direction: Direction of the Data 6 pin LCD_D5_Direction: Direction of the Data 5 pin LCD_D4_Direction: Direction of the Data 4 pin LCD_RS_Direction: Direction of the Register Select pin LCD_EN_Direction: Direction of the Enable signal pin must be defined before using this function.</p>
Example	<pre>// Lcd pinout settings sbit LCD_RS at RB4_bit; sbit LCD_EN at RB5_bit; sbit LCD_D7 at RB3_bit; sbit LCD_D6 at RB2_bit; sbit LCD_D5 at RB1_bit; sbit LCD_D4 at RB0_bit; // Pin direction sbit LCD_RS_Direction at TRISB4_bit; sbit LCD_EN_Direction at TRISB5_bit; sbit LCD_D7_Direction at TRISB3_bit; sbit LCD_D6_Direction at TRISB2_bit; sbit LCD_D5_Direction at TRISB1_bit; sbit LCD_D4_Direction at TRISB0_bit; ... Lcd_Init();</pre>

Table 2.1.1 Lcd_init Function description

Prototype	void Lcd_Out(char row, char column, char *text);
Returns	Nothing.
Description	<p>Prints text on Lcd starting from specified position. Both string variables and literals can be passed as a text.</p> <p>Parameters :</p> <p>row: starting position row number</p>

	column: starting position column number text: text to be written
Requires	The Lcd module needs to be initialized. See Lcd_Init routine.
Example	// Write text "Hello!" on Lcd starting from row 1, column 3: Lcd_Out(1, 3, "Hello!");

Table 2.1.2 Lcd_Out Function description

Prototype	void Lcd_Cmd(char out_char);
Returns	Nothing.
Description	Sends command to Lcd. Parameters : out_char: command to be sent Note : Predefined constants can be passed to the function, see Available Lcd Commands .
Requires	The Lcd module needs to be initialized. See Lcd_Init table.
Example	// Clear Lcd display: Lcd_Cmd(_LCD_CLEAR);

Table 2.1.3 Lcd_Cmd Function description

Available Lcd Commands

Lcd Command	Purpose
_LCD_FIRST_ROW	Move cursor to the 1st row
_LCD_SECOND_ROW	Move cursor to the 2nd row
_LCD_THIRD_ROW	Move cursor to the 3rd row
_LCD_FOURTH_ROW	Move cursor to the 4th row
_LCD_CLEAR	Clear display
_LCD_RETURN_HOME	Return cursor to home position, returns a shifted display to its original position. Display data RAM is unaffected.
_LCD_CURSOR_OFF	Turn off cursor
_LCD_UNDERLINE_ON	Underline cursor on
_LCD_BLINK_CURSOR_ON	Blink cursor on
_LCD_MOVE_CURSOR_LEFT	Move cursor left without changing display data RAM
_LCD_MOVE_CURSOR_RIGHT	Move cursor right without changing display data RAM
_LCD_TURN_ON	Turn Lcd display on
_LCD_TURN_OFF	Turn Lcd display off
_LCD_SHIFT_LEFT	Shift display left without changing display data RAM
_LCD_SHIFT_RIGHT	Shift display right without changing display data RAM

Table 2.1.4 LCD commands

2.6.4 SPI Library

The mikroC PRO for PIC provides a library for comfortable with SPI work in Master mode. The Library routines are listed below:

Prototype	void SPIx_Init();
Returns	Nothing.
Description	This routine configures and enables SPI module with the following settings:

	<p>master mode clock Fosc/4 clock idle state low data transmitted on low to high edge input data sampled at the middle of interval</p> <p>Note :</p> <p>PIC16F170x/171x family specifics : SPI1 module is initialized on the RC3, RC4, RC5 pins (SCK, SDI, SDO, respectively). PIC18FxxJ94 family specifics : SPI1 module is initialized on the RC3, RC4, RC5 pins (SCK, SDI, SDO, respectively). SPI2 module is initialized on the RD6, RD5, RD4 pins (SCK, SDI, SDO, respectively).</p>
Requires	You need PIC MCU with hardware integrated SPI.
Example	// Initialize the SPI1 module with default settings <pre>SPI1_Init();</pre>

Table 2.1.5 Spi_Init Function description

Prototype	unsigned short SPIx_Read(unsigned short buffer);
Returns	Returns the received data.
Description	<p>Reads one byte from the SPI bus.</p> <p>Parameters :</p> <p>buffer: dummy data for clock generation (see device Datasheet for SPI modules implementation details)</p>
Requires	You need PIC MCU with hardware integrated SPI. SPI must be initialized and communication established before using this function. See SPIx_Init_Advanced or SPIx_Init .
Example	<pre>short take, buffer; ... take = SPI1_Read(buffer);</pre>

Table 2.1.6 Spi_Read Function description

Prototype	void SPIx_Write(unsigned short data_);
Returns	Nothing.
Description	<p>Writes byte via the SPI bus.</p> <p>Parameters :</p> <p>wrdata: data to be sent</p>
Requires	You need PIC MCU with hardware integrated SPI. SPI must be initialized and communication established before using this function. See SPIx_Init_Advanced or SPIx_Init .
Example	<pre>// write a byte to the SPI bus char buffer; ... SPI1_Write(buffer);</pre>

Table 2.1.7 Spi_Write Function description

2.6.5 UART Library

The mikroC PRO for PIC UART Library provides comfortable work with the Asynchronous (full duplex) mode.

Prototype	<code>void UARTx_Init(const unsigned long baud_rate);</code>
Returns	Nothing.
Description	<p>Configures and initializes the UART module. The internal UART module module is set to: receiver enabled transmitter enabled frame size 8 bits 1 STOP bit parity mode disabled asynchronous operation Parameters : baud_rate: requested baud rate Refer to the device data sheet for baud rates allowed for specific Fosc.</p> <p>Note : Calculation of the UART baud rate value is carried out by the compiler, as it would produce a relatively large code if performed on the library level. Therefore, compiler needs to know the value of the parameter in the compile time. That is why this parameter needs to be a constant, and not a variable. PIC16F170x/171x family specifics : UART1 module is initialized on the RC4 and RC5 pin (TX and RX, respectively). PIC18FxxJ94 family specifics : UART1 module is initialized on the RC6 and RC7 pin (TX and RX, respectively). UART2 module is initialized on the RG1 and RG2 pin (TX and RX, respectively). UART3 module is initialized on the RE1 and RE0 pin (TX and RX, respectively). UART4 module is initialized on the RC5 and RC4 pin (TX and RX, respectively).</p>
Requires	You'll need PIC MCU with hardware UART.
Example	<code>// Initialize hardware UART1 and establish communication at 9600 bps UART1_Init(9600);</code>

Table 2.1.8 Uart_Init Function description

Prototype	<code>char UARTx_Data_Ready();</code>
Returns	1 if data is ready for reading 0 if there is no data in the receive register
Description	Use the function to test if data in receive buffer is ready for reading.
Requires	MCU with the UART module.

	The UART module must be initialized before using this routine. See the UARTx_Init routine.
Example	<pre>// If data is ready, read it: if (UART1_Data_Ready() == 1) { receive = UART1_Read(); }</pre>

Table 2.1.9 Uart_Data_Ready Function description

Prototype	char UARTx_Tx_Idle();
Returns	1 if the data has been transmitted 0 otherwise
Description	Use the function to test if the transmit shift register is empty or not.
Requires	UART HW module must be initialized and communication established before using this function. See UARTx_Init .
Example	<i>// If the previous data has been shifted out, send next data:</i> <pre>if (UART1_Tx_Idle() == 1) { UART1_Write(_data); }</pre>

Table 2.1.9 Uart_Tx_Idle Function description

Prototype	char UARTx_Read();
Returns	Returns the received byte.
Description	Function receives a byte via UART. Use the function UARTx_Data_Ready to test if data is ready first.
Requires	MCU with the UART module. The UART module must be initialized before using this routine. See UARTx_Init routine.
Example	<i>// If data is ready, read it:</i> <pre>if (UART1_Data_Ready() == 1) { receive = UART1_Read(); }</pre>

Table 2.2.0 Uart_Read Function description

Prototype	void UARTx_Write(char data_);
Returns	Nothing.
Description	The function transmits a byte via the UART module. Parameters : _data: data to be sent
Requires	MCU with the UART module. The UART module must be initialized before using this routine. See UARTx_Init routine.
Example	unsigned char _data = 0x1E; ... UART1_Write(_data);

Table 2.2.1 Uart_Write Function description

2.6.6 USB Library

USB Library contains HID routines that support HID class devices, and also the generic routines that can be used with vendor specified drivers.

Library Routines:

Prototype	<code>void HID_Enable(char *readbuff, char *writebuff);</code>
Description	Enables USB HID communication.
Parameters	readbuff: Read Buffer. writebuff: Write Buffer. These parameters are used for HID communication.
Returns	Nothing.
Requires	Nothing.
Example	<code>HID_Enable(&readbuff,&writebuff);</code>
Notes	This function needs to be called before using other routines of USB HID Library.

Table 2.2.2 `Hid_Enable` Function description

Prototype	<code>char HID_Read(void);</code>
Description	Receives message from host and stores it in the Read Buffer.
Parameters	None.
Returns	If the data reading has failed, the function returns 0. Otherwise, it returns number of characters received from the host.
Requires	USB HID needs to be enabled before using this function. See <code>HID_Enable</code> .
Example	<code>// retry until success</code> <code>while(!HID_Read())</code> <code>;</code>
Notes	None.

Table 2.2.3 `Hid_Read` Function description

Prototype	<code>char HID_Write(char *writebuff, char len);</code>
Description	Function sends data from Write Buffer writebuff to host.
Parameters	writebuff: Write Buffer, same parameter as used in initialization; see <code>HID_Enable</code> . len: specifies a length of the data to be transmitted.
Returns	If the data transmitting has failed, the function returns 0. Otherwise, it returns number of transmitted bytes.
Requires	USB HID needs to be enabled before using this function. See <code>HID_Enable</code> .
Example	<code>// retry until success</code> <code>while(!HID_Write(&writebuff,64))</code> <code>;</code>
Notes	Function call needs to be repeated as long as data is not successfully sent.

Table 2.2.4 `Hid_Write` Function description

Prototype	<code>void HID_Disable(void);</code>
Description	Disables USB HID communication.
Parameters	None.

Returns	Nothing.
Requires	USB HID needs to be enabled before using this function. See HID_Enable .
Example	HID_Disable();
Notes	None.

Table 2.2.5 [Hid_Disable](#) Function description

Prototype	void USB_Interrupt_Proc(void);
Description	This routine is used for servicing various USB bus events. Should be called inside USB interrupt routine.
Parameters	None.
Returns	Nothing.
Requires	Nothing.
Example	<pre>void interrupt() { USB_Interrupt_Proc(); }</pre>
Notes	Do not use this function with USB_Polling_Proc , only one should be used. To enable servicing through interrupt, USB_INTERRUPT constant should be set (it is set by default in descriptor file).

Table 2.2.6 [Usb_Interrupt_Proc](#) Function description

Prototype	void USB_Polling_Proc(void);
Description	This routine is used for servicing various USB bus events. It should be periodically, preferably every 100 microseconds.
Parameters	None.
Returns	Nothing.
Requires	Nothing.
Example	<pre>while(1) { USB_Polling_Proc(); kk = HID_Read(); if (kk != 0) { for(cnt=0; cnt < 64; cnt++) writebuff[cnt]=readbuff[cnt]; HID_Write(&writebuff,64); } }</pre>
Notes	Do not use this functions with USB_Interrupt_Proc . To enable servicing by polling, USB_INTERRUPT constant should be set to 0 (it is located in descriptor file).

Table 2.2.7 [Usb_Polling_Proc](#) Function description

2.7 USB Communication

Universal Serial Bus (USB) is a communications architecture that gives a personal computer (PC) the ability to interconnect a variety of devices using a simple four wire cable [8]. USB protocols can configure devices at startup or when they are plugged in at run time. These devices are broken into

various device classes. Each device class defines the common behavior and protocols for devices that serve similar functions. Some examples of USB device classes are:

Device Class	Example Device
Display	Monitor
Communication	Modem
Audio	Speakers
Mass storage	Hard drive
Human interface Data	glove

Table 2.2.8 Usb Communication Device Classes

2.7.1 The HID USB Class

The **HID** class consists primarily of devices that are used by humans to control the operation of computer systems. Typical examples of **HID** class devices include Keyboards and pointing devices.

Information about a USB device is stored in segments of its ROM called descriptors. A USB/HID class device uses a corresponding **HID** class driver to retrieve and route all data. The routing and retrieval of data is accomplished by examining the descriptors of the device and the data it provides.

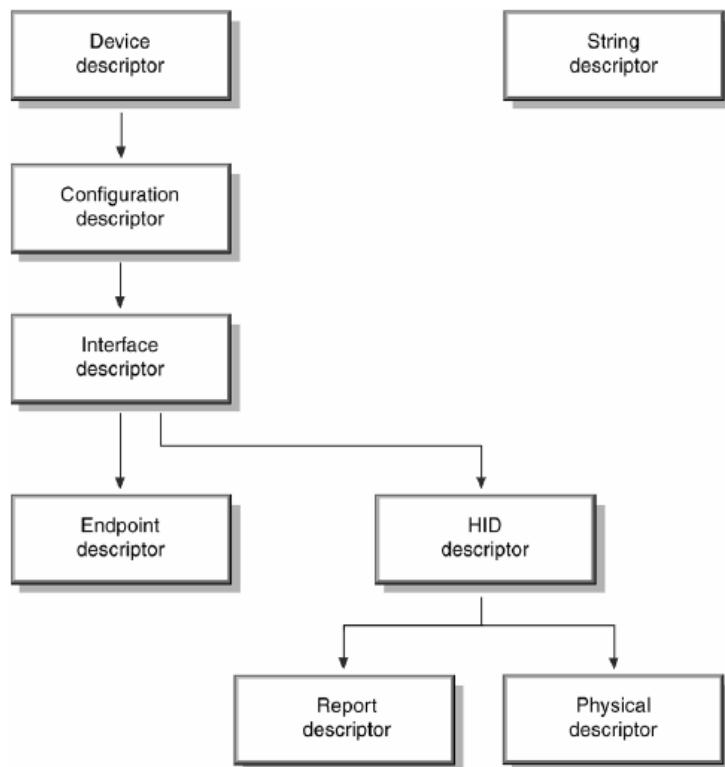


Figure 2.1.8 USB HID Descriptors

An interface descriptor can identify a device as belonging to one of a finite number of classes. The **HID** class device descriptor identifies which other HID class descriptors are present and indicates their sizes. A **Report** descriptor describes each piece of data that the device generates and what the data is actually

measuring. **Physical descriptor** sets are optional descriptors which provide information about the part or parts of the human body used to activate the controls on a device.

2.7.2 Interfaces

A **HID** class device communicates with the **HID** class driver using either the **Control** (default) pipe or an **Interrupt** pipe.

The **Control** pipe is used for:

- Receiving and responding to requests for USB control and class data.
- Transmitting data when polled by the **HID** class driver (using the **Get_Report** request).
- Receiving data from the host.

The **Interrupt** pipe are used for:

- Receiving asynchronous (unrequested) data from the device.
- Transmitting low latency data to the device.

2.7.3 Device Descriptor Structure

At the topmost level, a descriptor includes two tables of information referred to as the Device descriptor and the String descriptor. A standard USB Device descriptor specifies the Product ID and other information about the device. For example, Device descriptor fields primarily include:

- Class
- Subclass
- Vendor
- Product
- Version

Chapter 3: Desktop Application Design

This chapter covers the design of the desktop application. It details the design methodologies used and includes UML diagrams illustrating the systems implementation.

The design of the software will follow the Rational Unified Process Methodology through the following steps:

1. Requirements analysis
2. Implementation
3. Testing and evaluation

3.1 Software Specification for JPLC Editor

JPLC Editor is the name of the software to be written in Java that will allow users to download Ladder Logic programs to the PIC Microcontroller based PLC Emulator. In developing the Ladder Program, it will allow the user to:

- i. Create a Project and Save it as a project file.
- ii. Create Rungs, Tags, Branches and more basic functionality associated with creating a Ladder Diagram.
- iii. Verify and Compile the Ladder Diagram after which files associated with the Ladder Diagram are generated.
- iv. Download the program to the PLC via a USB Port.

3.1.1 Global Functionality

This section looks at the expected general functionality of the JPLC Editor application.

3.1.1.1 Save Project Functionality

This software is expected to allow the user to save their PLC project.

3.1.1.2 Help Functionality

The software should allow the user to access help information relating to the software.

3.1.1.3 Auto-fit Window Functionality.

The software should fit itself on the available screen resolution on the computer.

3.1.2 Ladder Project Functionality

This section looks at requirements concerning developing Ladder Diagrams.

3.1.2.1 Creating a New Project

Allow the user to create a project and fill in the following information:

- i. Project Name

ii. Project Working Directory

From here, the software should then create a project file and the source and termination rails.

3.1.2.2 Creating a Tag

The software should allow the user to create a variable asking for the following information:

- i. Tag name.
- ii. Tag type.

The tags shall be of one of the following types shown in the table below:

Data Type	Byte Size	Range
Bit	0	0-1
Unsigned Byte	1	0-255
Unsigned Int	2	0-32768

Table 3.1 Tag types

3.1.2.3 Creating Rungs and Branches

The software should allow the user to create rungs and branches by means of a drag and drop interface.

3.1.2.3 Adding Instructions to Rungs

The software should allow the user to pick an instruction from a list of instructions and drag it to the specified node on a rung.

3.1.2.4 Specifying Tags and values on Instructions

The software should allow the user to select a tag from a tag list and drag it over to a function block or instruction that supports the tag data type. It should also allow the user to enter values through the keyboard on the instructions.

3.1.2.5 Deleting Ladder Diagram Objects

Instructions on the Ladder Diagram should be selectable and respond to mouse clicks.

3.1.3 Compiler Functionality

This pertains to the converting of the ladder diagram to hexadecimal code. The compiler should:

- i. Check for errors in the user Ladder diagram after they have pressed the compile buttons.
- ii. After verification, the software should generate the files listed in the table below.

File Type	Extension	Description
Ladder Diagram file	ld	An instruction list for the ladder diagram
Asm file	asm	The asm code associated with the ladder diagram
Program file	pgm	The byte code for the user program and associated addresses for each byte.
Hex file	hex	The hex file for the program.

Table 3.2 Compilation generated files

3.1.4 Communication Functionality

The software should allow the user to download their ladder program to the PLC emulator and verify the program by reading it from the PLC.

3.2 UML Modelling

This section looks at the Unified Modelling Language used to develop the application.

3.2.1 Use case Diagram

Use cases are a formal way to capture and to express the interaction and dialog between system users (called actors) and the system itself (Conallen 2000, pp.122). Use cases are useful in describing the system functionality without detailing how it should do it, and are used in this project to clearly identify what is required before implementation begins.

The first task is to identify the actors of the system, of which there is one, the user of the system. Shown in the diagram below is the use case diagram that details the interaction of the actor with the system.

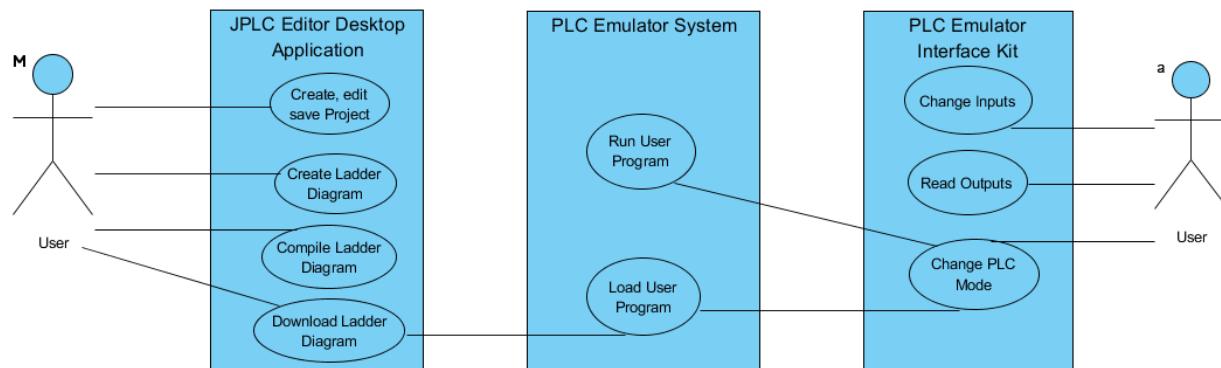


Table 3.1 System Use case diagram

3.2.2 Package Definition

A Package is a namespace used to group together elements that are semantically related and might change together. It allows to show different views of a system, for example, as multi-layered (aka multi-tiered) application [16].

The functionality will be divided into the following layers where each layer/ package performs an independent operation.

- User Interaction Layer
- Ladder Diagram Layer
- Compiler Layer
- Communications Layer
- Resources Layer
- Project Layer

The diagram below illustrates the package interaction

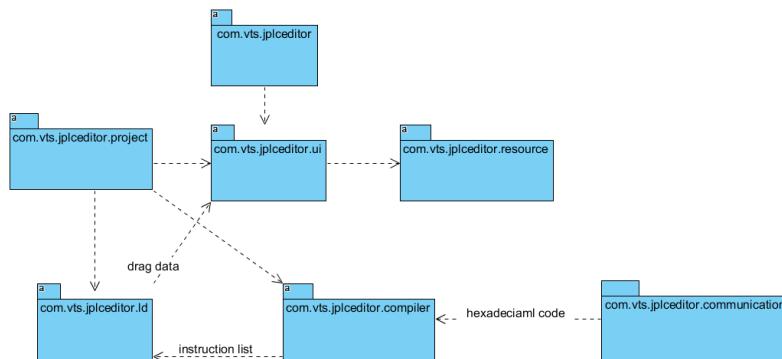


Figure 3.2 JPLC Editor package interaction

3.2.2.1 User Interaction Layer

- The Layer is under the package: `com.vts.jplceditor.ui`
- Functionality involves capturing user inputs and sending it to the required layer.
- Expected content will be mainly JavaFX FXML files and their controller classes.

3.2.2.2 Ladder Diagram Layer

- The Layer is under the package: `com.vts.jplceditor.id`
- Functionality involves drawing the ladder diagram and layout management for the ladder diagram components.
- Expected subclasses should divide the package into the ladder diagram components, nodes and instructions.

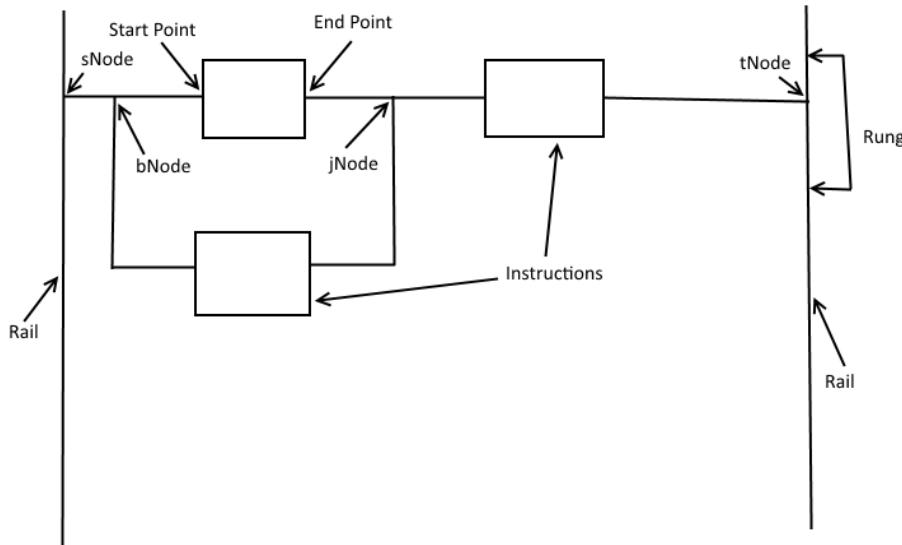


Figure 3.3 Ladder Diagram components

A Ladder Diagram (LD) presented as a diagraph can be considered as a tree with the root as a single entity left bar. Taking the left bar as a source node, it will have only one source node, the left bar to traverse the entire graph. It can have several sink nodes as the application allows, but these have out degree equal to zero and in degree equal to one, the source has in degree equal to zero and can have plenty of out degree depending on your application. Virtual nodes are required to represent the interconnection for parallel network. Virtual nodes have in degree or out degree greater than or equal to one. Given the source then we can begin to generate code by depth first traversal till we reach a virtual node.

This software will use the following notations for the ladder network:

- Source nodes will be denoted as sNODE
- terminal or sink node as tNODE
- virtual nodes as bNODE and jNode
- contact nodes or instruction nodes as iNODEs.
- There are also special node types these are isNODEs and isNODEs. These nodes are associated with contact blocks.

3.2.2.3 Compiler Layer

- The Layer is under the package: com.vts.jplceditor.compiler
- Functionality involves converting an instruction list into assembly and finally hexadecimal code.

3.2.2.4 Communication Layer

- The Layer is under the package: com.vts.jplceditor.communication
- Functionality involves transmitting the ladder program to the PLC Emulator.

3.2.2.5 Resources Layer

- The Layer is under the package: com.vts.jplceditor.resource
- Functionality involves organizing system resources like icons and fxml files.

3.2.2.6 Project Layer

- The Layer is under the package: com.vts.jplceditor.project
- Functionality involves organizing the ladder diagram information.

3.2.3 Software Functionality Sequence Specification

This section seeks to fully clarify the software specification and operation based on previously designed UML diagrams.

3.2.3.1 Creating a New Project

- User clicks the create new project button.
- Create New project window appears asking for:
 - Project name.
 - Project working directory
- A folder with the project name is created in the working directory.
- An xml file for the project is created.

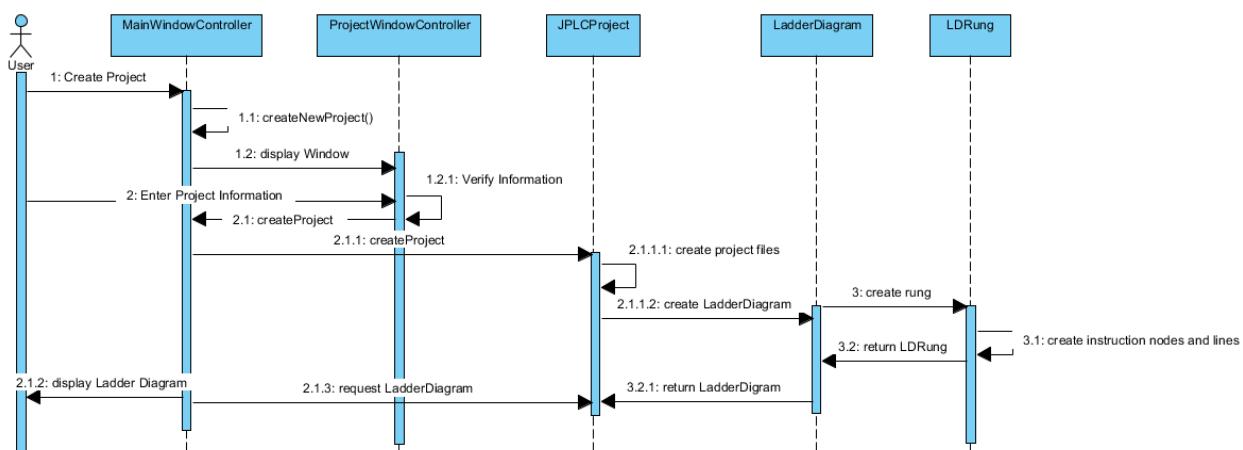


Figure 3.4 Creating a new project sequence diagram

3.2.3.2 Opening a Project

- User Clicks open project button.
- File Browser window is shown.
- User selects valid project file.
- System accepts project file and creates a project with the information specified in the project file.

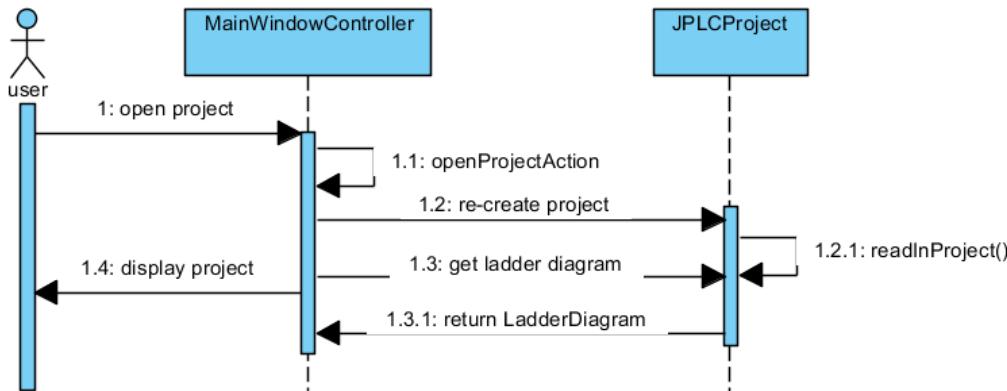


Figure 3.5 Opening a project sequence diagram

3.2.3.4 Creating a Rung

- User clicks create rung button.
- Create rung window appears, asking the user for the rung number.
- User enters a valid rung number.
- System creates the rung.

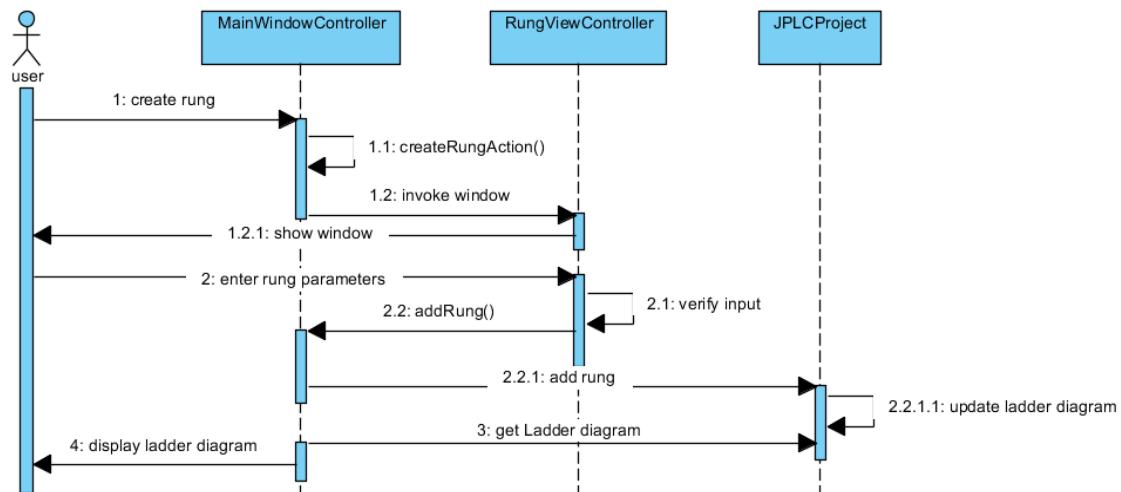


Figure 3.7 Creating a rung sequence diagram

3.2.3.3 Creating a Tag

- User clicks create a tag button.
- Create new tag window appears.
- User enters valid tag information.
- System accepts tag and adds it to the tag list.

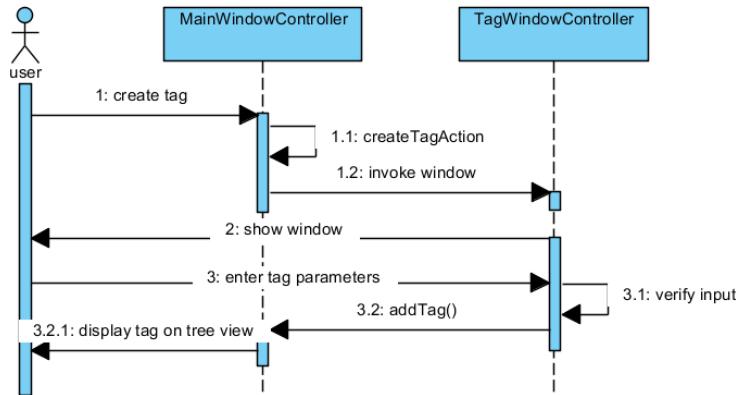


Figure 3.6 Creating a tag sequence diagram

3.2.3.5 Dragging data onto a Ladder diagram

- User clicks and drags on either a tag or instruction.
- The system gets information about the dragged item and puts it onto the drag board clip board.
- User drags the tag or instruction onto the ladder diagram. System uses information on the clip board to determine where to accept and reject the dragged item.
 - If the dragged item is an instruction, it can only be dropped onto an instruction node.
 - If the dragged item is a tag, the tag data type will determine where the tag will be dropped on an instruction.
- User dropped valid data onto the ladder diagram.
 - If the user drops on an instruction node, the system will use the data on the clip board to determine which instruction to place on the instruction node.
 - If the user drops a tag, the instruction should display the name of the tag.

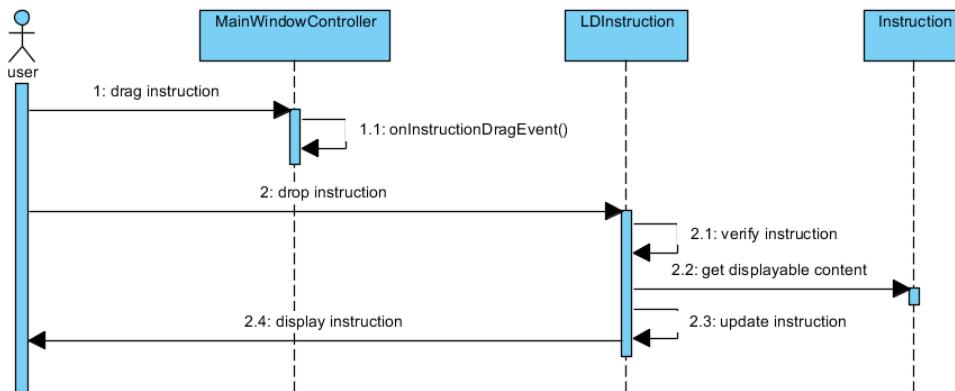


Figure 3.8 Dragging an instruction sequence diagram

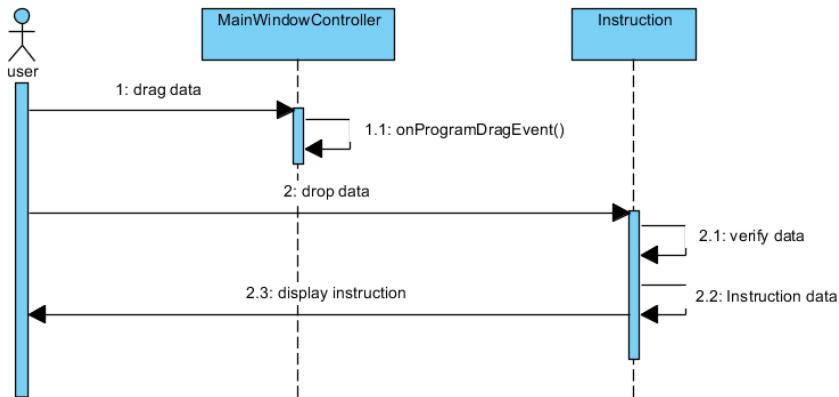


Figure 3.9 Dragging a tag sequence diagram

3.2.3.6 Compile Program

- i. User clicks compile button.
- ii. System scans the ladder diagram for errors and displays the status in a status field.
- iii. If no errors are found, System first scans the ladder diagram then generates a rung list for the ladder diagram where each rung contains an instruction list.
- iv. System then uses the rung list to generate an instruction list.
- v. The system then assigns addresses in PLC RAM for the tags and data associated with each instruction.
- vi. The System then generates an assembly instruction list.
- vii. The system then converts the assembly instruction list to 16-bit hex code and fills it into a byte array.
- viii. The system then generates the four ladder program files specified earlier.

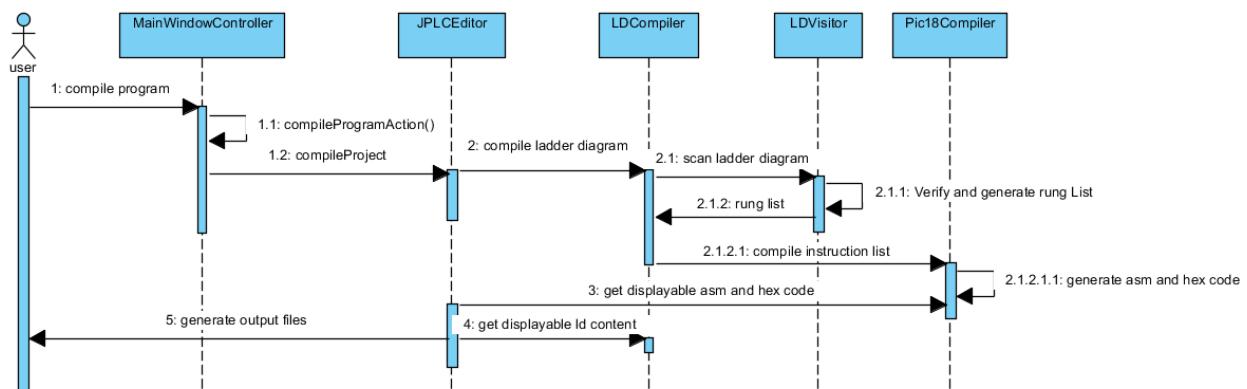


Figure 3.1.0 Compiling Ladder diagram sequence diagram

3.2.3.7 Download Program

- i. User clicks download button.
- ii. System checks if the ladder diagram is compiled.
- iii. If Ladder diagram is compiled, system obtains the byte array for the hex code from the compiled program.
- iv. The system then divides the number of bytes in the byte array by 32, to determine the number of programming sessions.
- v. The system then encodes the divided byte arrays into separate PLC messages which will be programmed at a specific address.
- vi. The system then performs a bulk transfer of the encoded data via USB to the PLC emulator.
- vii. The system then reads the PLC program from the PLC to verify the programming.
- viii. The system displays the whole operation on the status field of the GUI to the user.

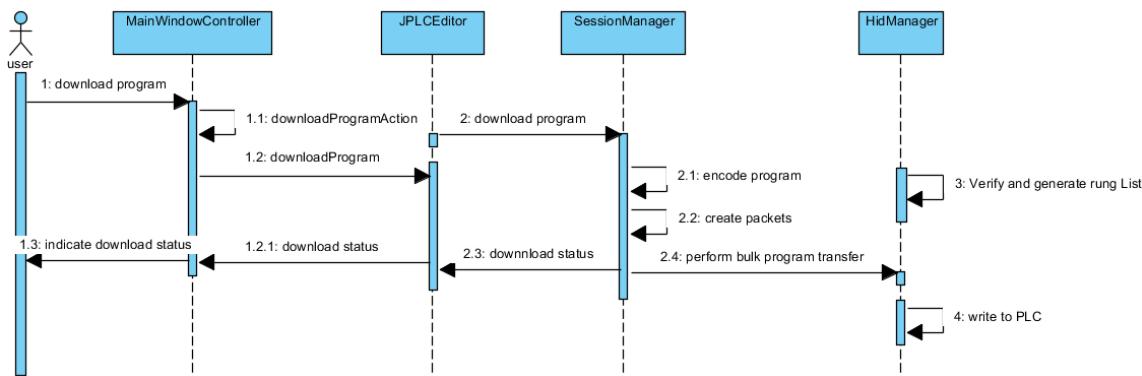


Figure 3.1.1 Downloading user program sequence diagram

3.3 USB Communication Protocol design

This section covers the design of the protocol to be used by the JPLC editor application to communicate with the PLC emulator. The design follows two simple steps:

1. Communication requirements.
2. Protocol implementation

3.3.1 Protocol Communication Requirements

The protocol is expected to allow the desktop application to perform the following processes on the emulator:

1. Write to the microcontroller flash memory.
2. Read the microcontroller flash memory.
3. Write to the microcontroller data memory.
4. Read the microcontroller data memory.
5. Synchronize PLC emulator and desktop application.

3.3.2 Protocol implementation

This section covers the implementation of the communication protocol that meets the needs of the specification outlined in section 3.4.1. The implementation follows the following steps:

1. Protocol format
2. Protocol representation
3. Protocol sessions

3.3.2.1 Protocol Format

Since the microcontroller will also be communicating with the operating system using the same USB channel, it is necessary to have a protocol format different from the one used by the operating system.

3.3.2.1.1 DF1 Protocol format

For starters, the protocol format was borrowed from the DF1 protocol used in Allen Bradley PLCs over the RS232 channel, with the hope of fulfilling the protocol specification if the Project has the enough time [18].

The data link Layer protocol format is of the form:

DLE STX xxxx DLE ETX BCC/CRC

Where:

- DLE STX is the sender symbol that separates the data link drop header from the data.
- Xxxx is the application layer data to be transmitted by the protocol.
- DLE ETX is the sender symbol that terminates a message.
- BCC/CRC are used for error checking.

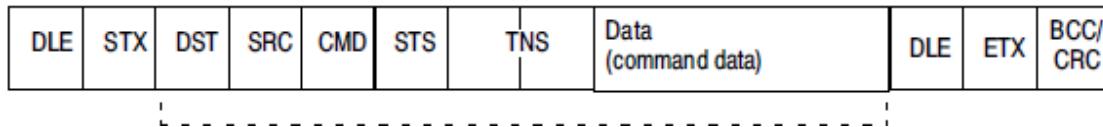


Figure 3.1.2 DF1 protocol message frame

The complete message frame with the application layer data is shown in figure 3.1.2. Here:

- DST and SRC represent the destination and source nodes.
- CMD represents the command to be performed at the destination.
- STS indicates the sender status.
- TNS represents the transmission number.

3.4.2.1.2 UHB protocol format

UHB protocol is the mikroElectronica USB HID Bootloader (UHB) protocol implementation [17]. The protocol follows the format:

<STX><CMD_CODE><ADDRESS><COUNT> <DATA>

Where:

- STX - Command start delimiter. Length: 1 byte. Mandatory.

- CMD_CODE - Command to be performed. Length: 1 byte. Mandatory.
- ADDRESS - Address field. Flash start address for CMD_CODE command operation. Length: 4 bytes. Optional (command specific).
- COUNT - Count field. Amount of data/blocks for CMD_CODE command operation. Length: is Optional (command specific).
- DATA - Data array.

3.4.1.3 JPLCEditor Protocol Format

The developed protocol format follows the following format:

<DLE><STX><CMD><ADDRESS><LENGTH> <DATA[LENGTH]><DLE><ETX>

- DLE - Message start delimiter. Length: 1 byte. Mandatory.
- STX - Message start delimiter. Length: 1 byte. Mandatory.
- CMD - Command. Length: 1 byte. Mandatory.
- ADDRESS - Address field. Start address for CMD command operation. Length: 2 bytes. Optional (command specific).
- LENGTH - Length field. Amount of data/blocks for CMD command operation. Length: 1 byte. Optional (command specific).
- DATA - Data array. Length: Length bytes. Optional (command specific).
- ETX - Message end delimiter. Length: 1 byte. Mandatory.

This is the message format for the desktop application which the emulator is expected to interpret. The emulator on the other hand acts as a slave to the desktop application hence its response format is expected to differ. The response is either data or an acknowledgement message. The acknowledgement message follows the following format:

<DLE><CMD ><STATUS><TRANSACTIOD ID>

- DLE - Response start delimiter.
Length: 1 byte. Mandatory.
- CMD - The Command we want to acknowledge.
Length: 1 byte. Mandatory.
- STATUS - The status of the command execution.
Length: 1 byte. Mandatory.
- TRANSACTION ID - Command execution number. Field is incremented every time an acknowledgement is sent.
Length: 2 bytes. Mandatory.

3.3.2.2 Protocol Representation

The data link layer protocol will use the DF1 Full duplex representation [17].

Abbreviation	Hexadecimal value
DLE	0x10
STX	0x02
ETX	0x03

Table 3.5 JPLC Editor Data Link Layer protocol representation

The commands to be executed will be one of the ones listed in table 3.6

Abbreviation	Hexadecimal value
cmdFREE	0x0
cmdWRITE_FLASH	0x1
cmdREAD_FLASH	0x2
cmdWRITE_RAM	0x3
cmdREAD_RAM	0x4
cmdERASE	0x5
cmdCONNECT	0x6

Table 3.6 JPLC Editor command representation

The status field of the response message will be one of the following listed in table 3.7

Status Code	Description
0x06	Command was successfully executed
0x0F	Command was not executed

Table 3.7 JPLC Editor status codes

3.3.2.3 Protocol Sessions

This section illustrates all of the possible communication sessions between the desktop application and the PLC emulator.

3.3.2.3.1 Synchronizing desktop application and PLC

This message frame is independent of the other application layer protocol data except the command to be executed.

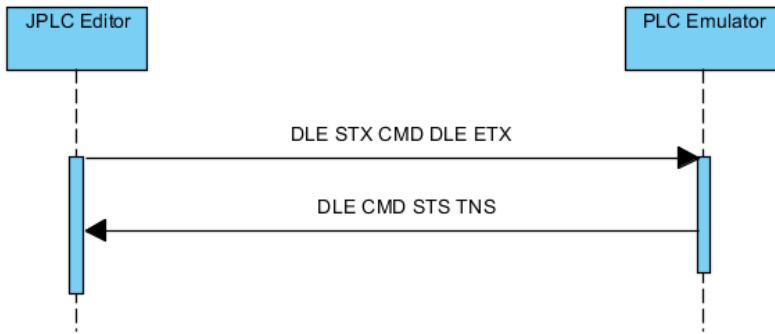


Figure 3.1.3 Synchronizing desktop application with PLC emulator session

The PLC emulator responds by indicating in the status field whether it is ready to synchronize with the application.

3.4.2.3.2 Writing to the PLC emulator

The figure below illustrates the communication session for writing to either the RAM or flash memory and for performing an erase operation. However, for the command “cmdERASE” which erases flash memory, the length field does not indicate the number of bytes to erase but the number of 64 byte blocks to erase.

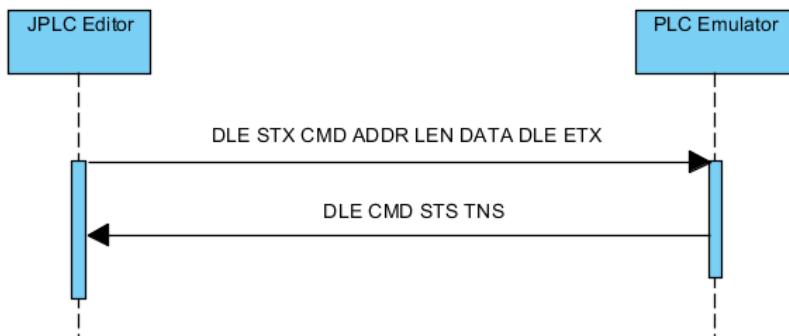


Figure 3.7 JPLC Editor write to PLC emulator session

In either case, the PLC emulator responds in the status field indicating whether the command was executed.

3.4.2.3.3 Reading from the PLC emulator

The figure below illustrates the communication session for reading from the PLC emulator. In this case, the only response from the emulator is the data to be read.

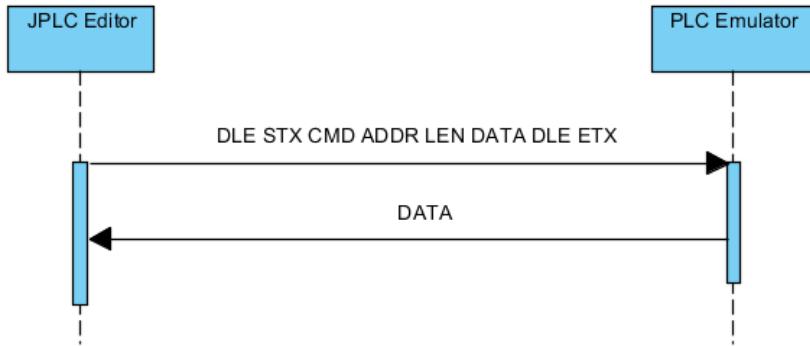


Figure 8 JPLC Editor read from PLC emulator session

Chapter 4: Hardware Design and Testing

This chapter covers the analysis and design of the physical PLC Emulator. It details the design methodologies used and includes flow chart diagrams.

4.1 Microcontroller Design

This section covers the development of the firmware for the operating system microcontroller and the I/O interface microcontroller.

4.1.1 Pin assignment

The I/O interface microcontroller is expected to have 8 digital inputs on PORD, 8 digital outputs on PORTB, 8 analogue inputs on analogue channels 0 to 7 and a single analogue output on the MCP 4921 which is connected via an SPI bus.

Pin Description	Pin assigned	Pin Number
Device Reset	RE3,MCLR	1
Oscillator pin 1	RA7	13
Oscillator pin2	RA6	14
Positive Power supply	Vdd	32, 12
Negative Power supply	Vss	31,11
8 digital Inputs	PortD	19,20,21,22,27,28,29,30
8 digital outputs	PortB	33,34,35,36,36,38,39,40
Analog input 0	RA0,AN0	2
Analog input 1	RA1,AN1	3
Analog input 2	RA2,AN2	4
Analog input 3	RA3,AN3	5
Analog input 4	RA5,AN4	7
Analog input 5	RE0,AN5	8
Analog input 6	RE1,AN6	9
Analog input 7	RE2,AN7	10
MCP 4921 CS	RC2	17
MCP 4921 SDI	RC0	15
MCP 4921 SCK	RC1	16
Serial Bus RX	RC7,RX	26
Serial Bus TX	RC6,TX	25
PLC Mode	RC3	18

Table 4.1 pin assignment for I/O interface microcontroller

The user program microcontroller is expected to interface to an LCD to indicate PLC status, connect to the PC USB port via the inbuilt USB support and communicate with the I/O interface microcontroller via a UART bus.

Pin Description	Pin assigned	Pin Number
Device Reset	RE3,MCLR	1
Oscillator pin 1	OSC1	10
Oscillator pin2	RA6,OSC2	9
Positive Power supply	Vdd	20
Negative Power supply	Vss	19,8
LCD E	RB4	25
LCD RS	RB5	26
LCD D4	RB0	21
LCD D5	RB1	22
LCD D6	RB2	23
LCD D7	RB3	24
USB VOLTAGE REGULATOR	VUSB	14
USB D+	RC5	16
USB D-	RC4	15
Serial Bus RX	RC7,RX	18
Serial Bus TX	RC6,TX	17
PLC Mode	RA0	2

Table 4.2 pin assignment for the operating system microcontroller

Affordable PLC Emulator For Students by Vusumuzi Tshabangu N0110328Z

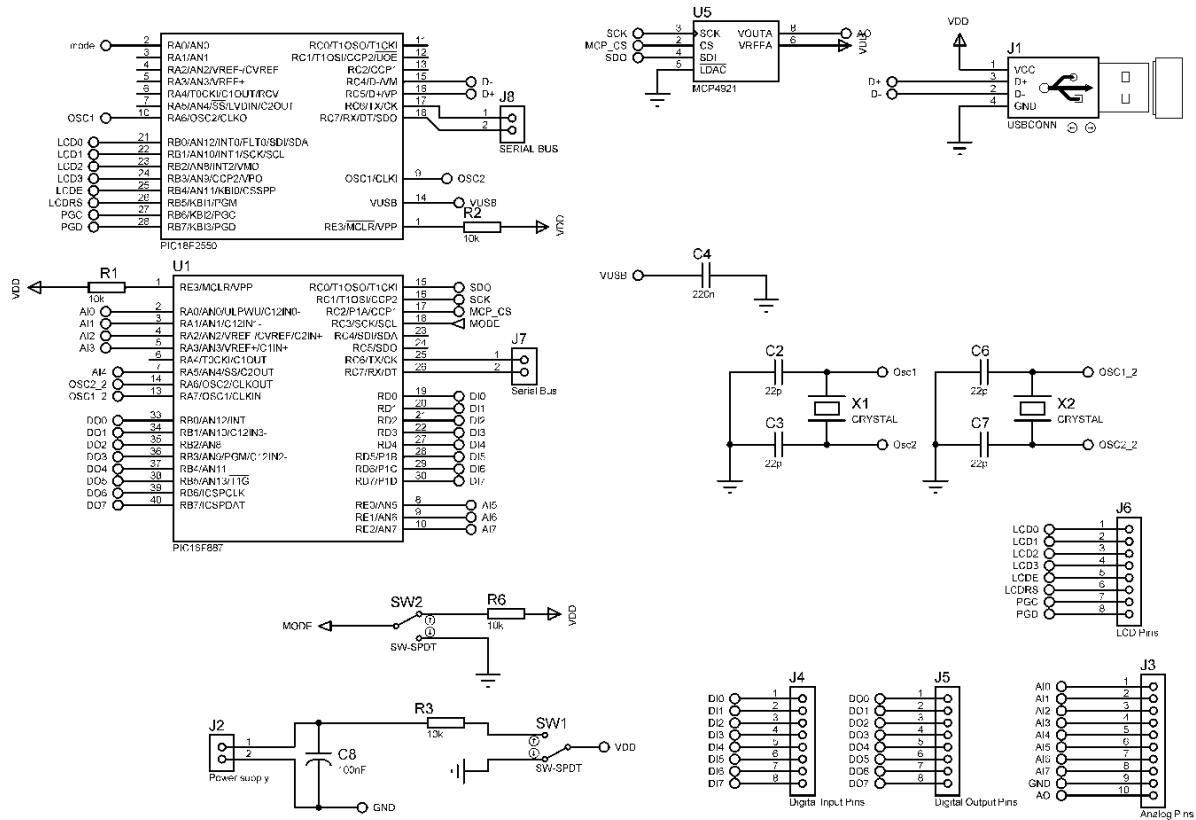


Figure 4.1 PLC emulator CPU board

4.1.2 User program microcontroller Flow Cart

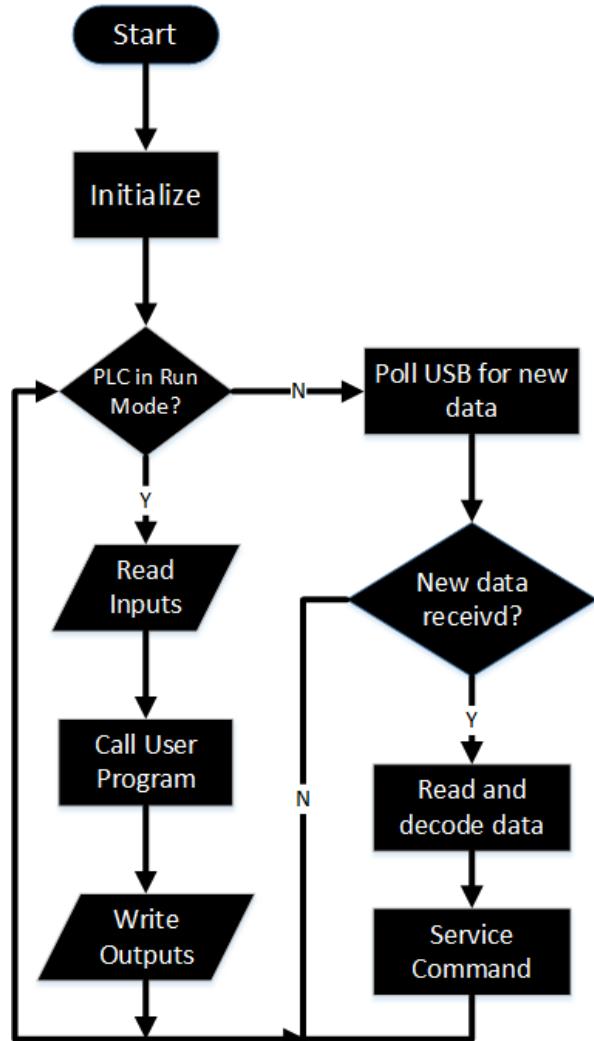


Figure 4.2 User program microcontroller flow chart

The initialization part of the flow chart involves configuring I/O port pin directions, and initializing the Lcd, USB and USART modules of the microcontroller. Then determine the state of the PLC by reading the plc mode pin then switching to the desired PLC mode.

In run mode, reading inputs involves sending a read request to the I/O interface microcontroller via the USART and reading in the response for all input bits. Calling the user program involve invoking the function which is placed at a memory location where the user program is expected to be located. Writing the outputs involves writing the outputs via the USART byte by byte.

In program mode, the USB module continually polls the input buffer to see if new data is received. If data is available, the data is decoded to make sure it follows the JPLC Editor protocol, after which the command in the protocol is determined and executed.

4.1.3 I/O Interface Microcontroller Flow chart

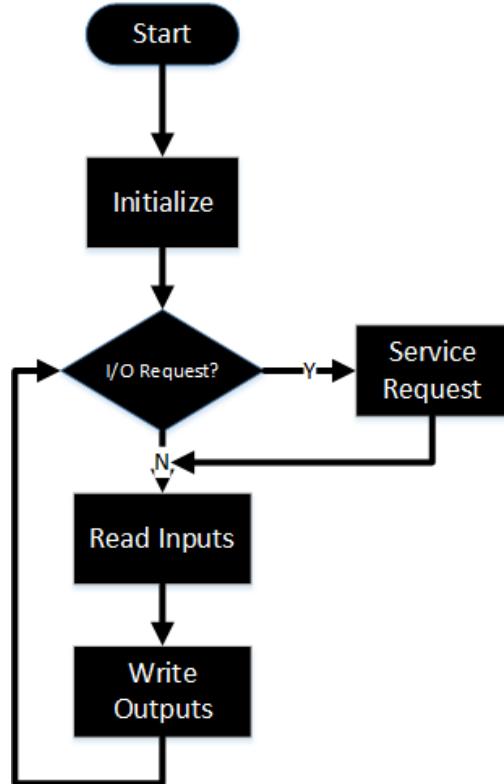


Figure 4.3 I/O interface microcontroller Flow chart

The initialization part involves configuring pin directions and pin usage as either analogue pins or digital pins. Then the SPI, ADC and USART modules are initialized.

The microcontroller then enters an endless loop of reading analogue and digital inputs and upon request from the user program microcontroller to either read from or write to this microcontroller, service that request via interrupt.

4.1.4 Program Memory Assignment

The Pic18F2550 has 32 Kbytes of Flash memory at addresses 000000h to 007FFFh. This memory is divided into 4 memory blocks and a boot block.

Memory Block	Address	Usage
Boot Block	000000h to 0007FFh	Operating System
Block 0	0008000h to 001FFFh	Operating System
Block 1	002000h to 003FFFh	User program
Block 2	004000h to 005FFFh	User program
Block 3	006000h to 007FFFh	User program

Table 4.3 User program microcontroller Program memory assignment

Hence, 8 Kbytes of flash memory at addresses 000000h to 001FFFh (8192) is assigned to the PLC operating system and 24Kbytes to the user program.

4.1.5 Data Memory Assignment

The Pic18F2550 has 2 Kbytes of RAM at addresses 000h to 7FFh. The memory space is divided into 16 banks 256 bytes each.

Bank	Address Range	Use
0	060h - OFFh	Operating System Data Variables, I/O image
1	100h – 1FFh	User program data image
2	200h – 2FFh	User program data image
3	300h – 3FFh	User program data image
4	400h – 4FFh	USB
5	500h – 5FFh	USB
6	600h – 6FFh	USB
7	700h – 7FFh	USB

Figure 9 User program microcontroller RAM assignment

4.1.5.1 Input and Output Image Table

The available inputs and outputs will be assigned the addresses listed below in RAM. These addresses will be accessible from the user program.

I/O image variable	Address
Analogue Input 0	0x60
Analogue Input 1	0x62
Analogue Input 2	0x64
Analogue Input 3	0x66
Analogue Input 4	0x68
Analogue Input 5	0x6A
Analogue Input 6	0x6C
Analogue Input 7	0x6E
Digital Inputs 0 to 7	0x70
Analogue Output 0	0x71
Digital Outputs 0 to 7	0x73

4.2 Opto-Isolator Design

This section covers the design of the circuit used to isolate the PLC microcontrollers from 24V DC.

4.2.1 Input Opto-isolator

The figure below shows a single channel using the CNY74-4 opto-isolator IC.

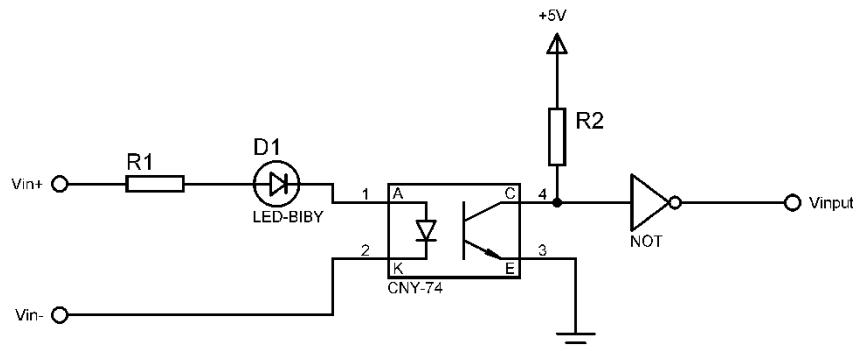


Figure 4.4 Digital Input opto-isolator circuit

Parameter	Rating
Opto Isolator Rated current	60mA
LED rated current	60mA
Opto isolator collector current	50mA
Opto isolator peak collector current	100mA
Opto isolator diode forward voltage	1.25V
LED forward voltage	1.2V

$$R_1 = \frac{24 - 1.25 - 1.2}{I_{R1}}$$

For $I_{R1} = 20\text{mA}$, $R_1 = 1\text{k ohm}$.

$$R_2 = \frac{5 - V_{CE}}{I_{R2}}$$

For $I_{R1} = 20\text{mA}$, $R_2 = 250\text{ ohm}$.

4.2.2 Output Opto-isolator

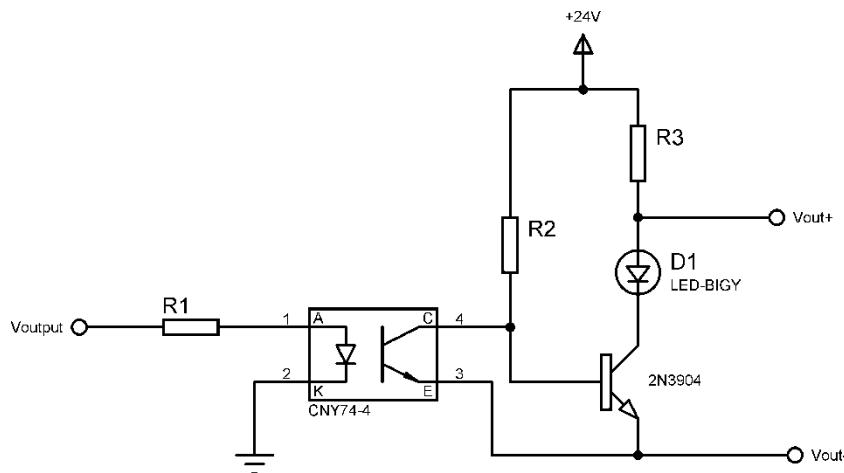


Figure 4.5 digital output opto-isolator circuit

Parameter	Rating
Transistor saturation collector current	50mA
Collector Emitter maximum saturation voltage	60V

$$R_1 = \frac{5 - 1.25}{I_{R1}}$$

For $I_{R1} = 25\text{mA}$, $R_1 = 150 \text{ ohm}$.

For switching purposes and $I_{out} \approx 25\text{mA}$, $R_3 = R_2 = 1\text{k ohm}$.

Chapter 5: Results and analysis

This chapter looks at the unit tests performed on the hardware and software to produce the final PLC Emulator System.

5.1 JPLC Editor Design Implementation and Testing

This section describes the processes implemented in code to produce the functionality of JPLC Editor.

5.1.1 Default Graphical User Interface

The JPLC Editor software was designed using the Netbeans IDE and was split into two projects. The first project was the JPLC Editor project library named JPLC Editor and consists of the Ladder diagram and Compiler layers or packages. It was then linked to the main project, which contains the main class JPLCEditor in the com.vts.jplceditor package. The main project named JPLC Editor-UI contains the User Interface Layer, the Resource Layer, the Communications Layer and the Project Layer.

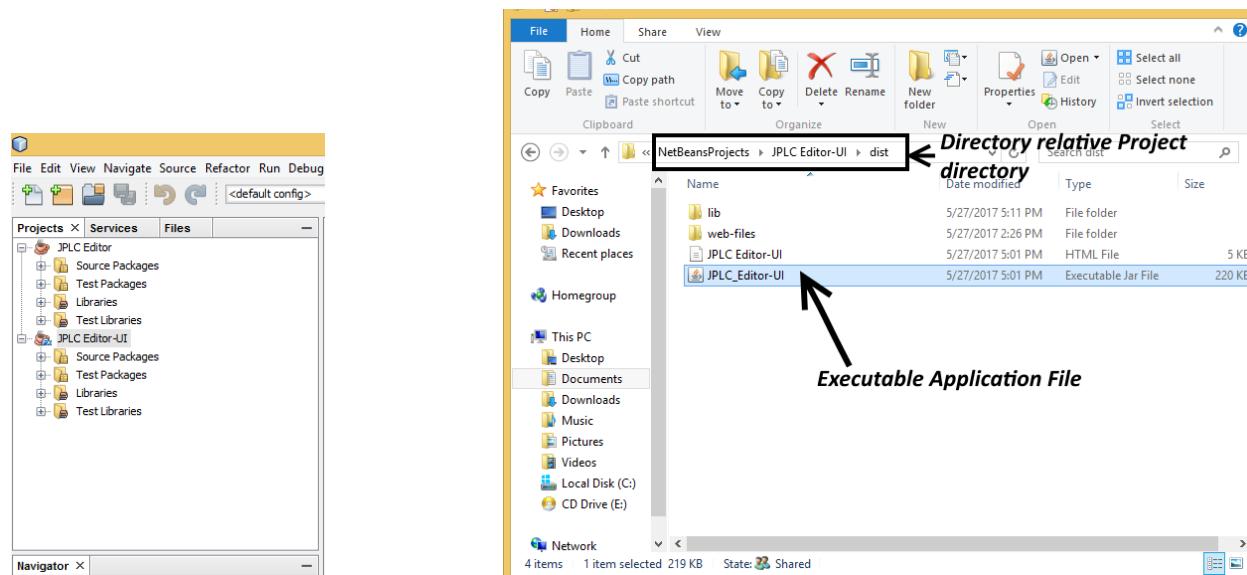


Figure 5.1 JPLC Editor Netbeans Projects and Executable file

Selecting the main project and running it on Netbeans or Launching the executable application file launches the default JPLC Editor window shown in figure 5.2. The menu bar contains all of the basic functionality under the following menu headings:

1. File Menu
 - a. Create new project
 - b. Open project
 - c. Save project
 - d. Exit
2. Edit Menu
 - a. Undo
 - b. Redo
 - c. Delete
3. Program Menu

- a. Save Program
- b. Create:
 - i. Tag
 - ii. Rung
 - iii. Program
- 4. Compiler Menu
 - a. Verify
 - b. Compile
- 5. Communications Menu
 - a. Download Program

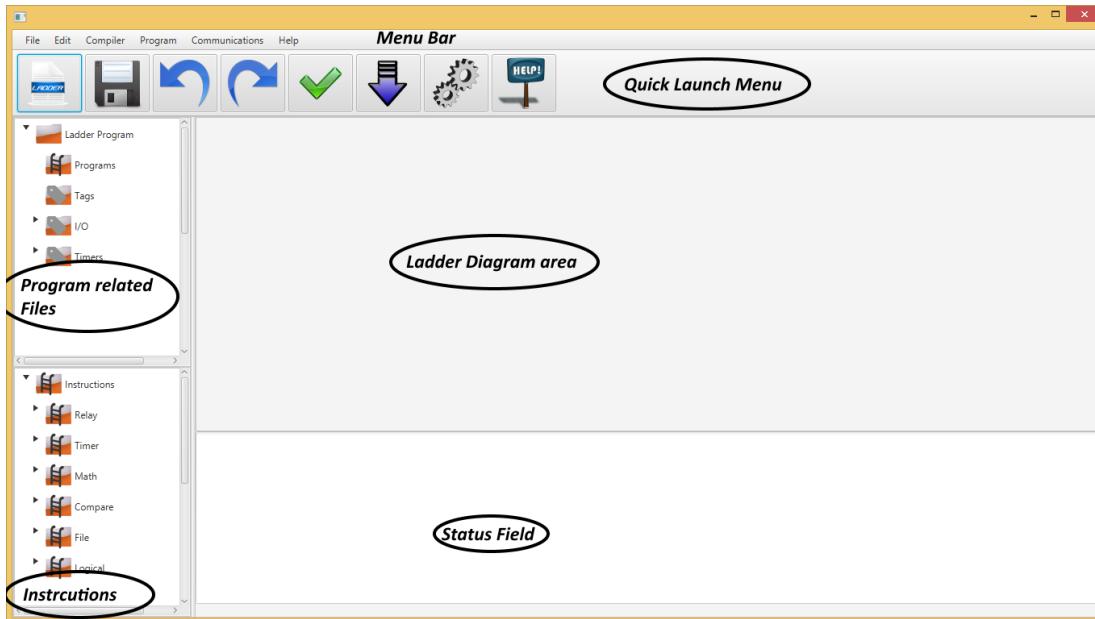


Figure 5.2 Default JPLC Editor Window

The Graphical User Interface (GUI) was designed using the JavaFX scene builder software [16]. It consists of a menu bar, tool bar, a side bar, a status field and the Ladder diagram view.

5.1.1.1 Implementation in code

The JPLC Editor main class in the `com.vts.jplceditor` package is the gateway to the application and launches as a JavaFX main class in the main method. The main method launches the primary Stage or window in the start method.

The application loads the FXML file identified by the url variable, which represents a Java URL object, pointing to the location of the main content that will be displayed in the primary window. The R class of the `com.vts.jplceditor.resource` package is a handy class that contains static string variables which contain the relative paths of the application's resources. In this case, the `WINDOW_MAIN` string variable is defined as:

```
public static String WINDOW_MAIN =
"/com/vts/jplceditor/ui/fxml/MainWindow.fxml";
```

In the R class.

```
public void start(Stage primaryStage) {
    url = getClass().getResource(R.WINDOW_MAIN);
    screenBounds = Screen.getPrimary().getVisualBounds();
    try {
        FXMLLoader loader = new FXMLLoader(url);
        root = loader.load();
        Scene scene = new Scene(root);
        primaryStage.setX(screenBounds.getMinX());
        primaryStage.setY(screenBounds.getMinY());
        primaryStage.setWidth(screenBounds.getWidth());
        primaryStage.setHeight(screenBounds.getHeight());
        primaryStage.setScene(scene);
        primaryStage.show();
    } catch (IOException ex) {
        Logger.getLogger(JPlcEditor.class.getName()).log(Level.SEVERE,
null, ex);
    }
}
```

When window an FXML window is loaded onto a JavaFX Stage or window, the *initialize()* of its Controller is executed which initializes any custom content. In this case, the controller is the MainWindowController class of the com.vts.jplceditor.ui.controller package.

5.1.2 Creating a new Project

A new project is created by clicking the New Ladder Button in the quick launch buttons menu or the New Project menu item from the menu bar. This displays the Create Project pop up window shown in figure 5.3 below.

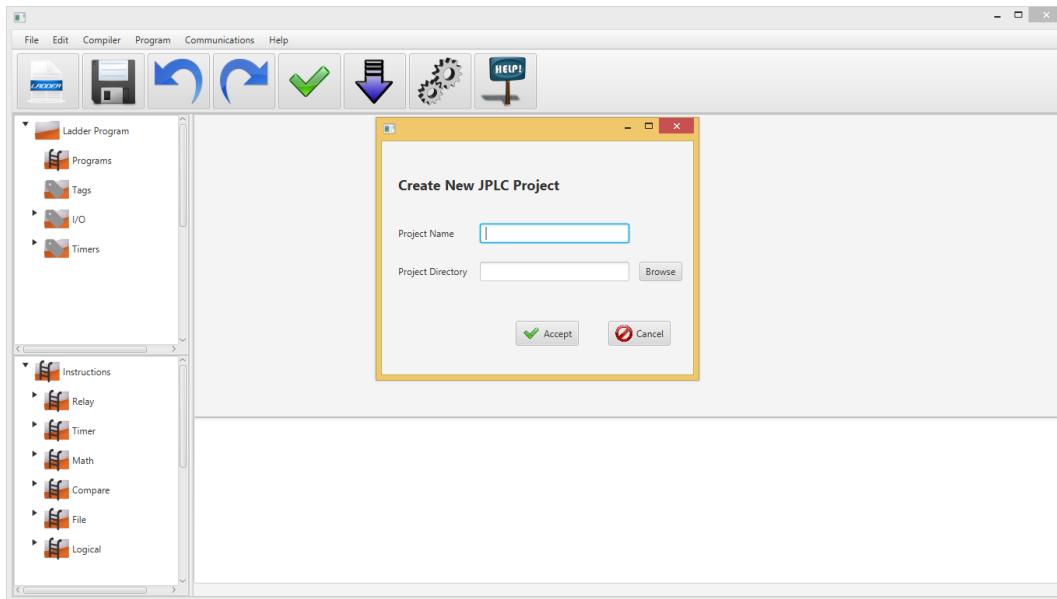


Figure 5.3 Create Project pop up window

Pressing the browse button displays a file chooser window that allows the user to select the project working directory which is illustrated in figure 5.4 below.

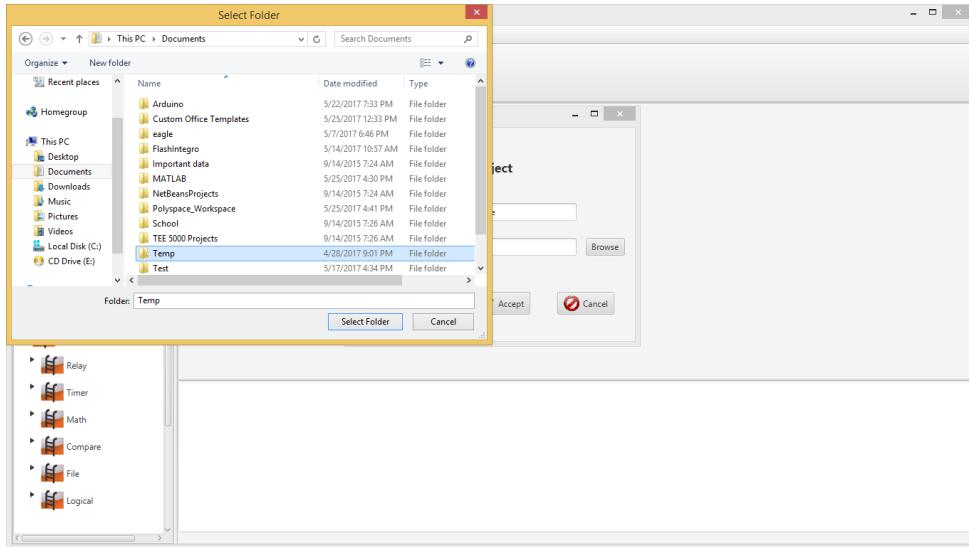


Figure 5.4 User browsing for the project working directory

Once the user chooses the working directory and presses the accept button, the project folder is created and the default ladder diagram.

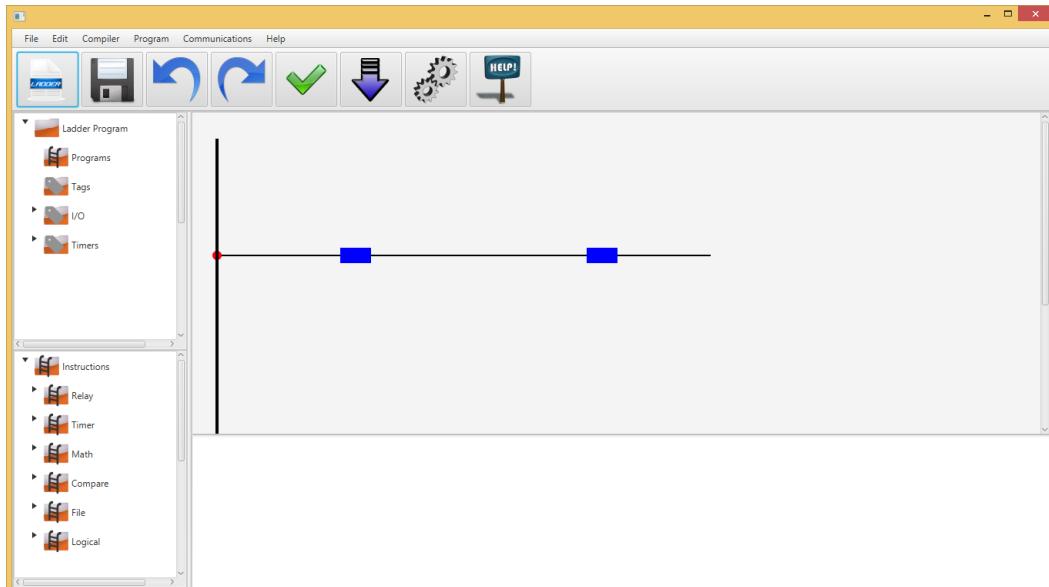


Figure 5.5 Default ladder diagram with instruction blocks created on project creation

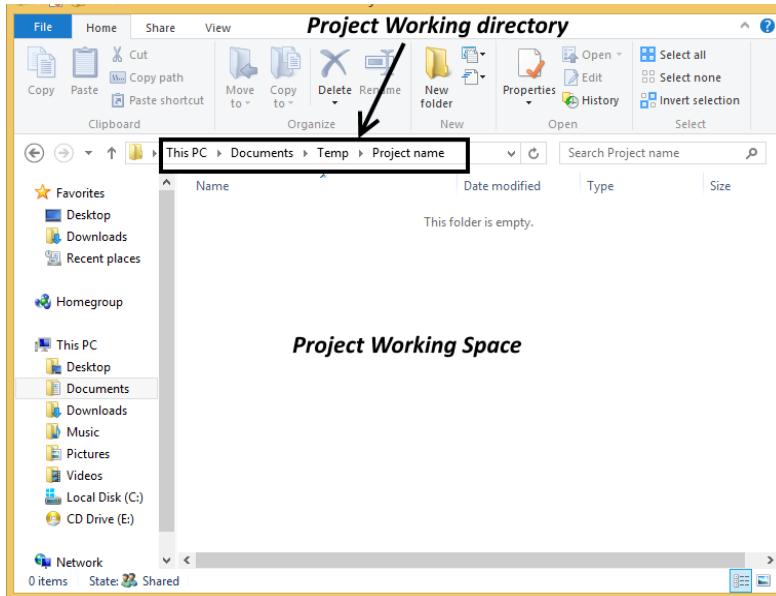


Figure 5.6 User project working folder

5.1.2.1 Implementation in code

Pressing either of the create project buttons results in the following code being executed in the MainWindowController class of the com.vts.jplceditor.ui.controller package:

```
@FXML  
private void createNewProject(ActionEvent event) {  
    url = getClass().getResource(R.WINDOW_PROJECT);  
    showPopup(Popup.ProjectWindow);  
}
```

The showPopup method displays a popup window depending on the enum parameter specified, in this case the Project Window. This displays the Create Project Window whose window controller is the ProjectWindowController class in the same controller package.

Pressing the browse button results in a file browser window appearing for an easier way to specify the working directory. This occurs when the browseButtonAction method is executed from the ProjectController class which is illustrated below:

```
@FXML  
private void browseButtonAction(ActionEvent event) {  
    DirectoryChooser chooser = new DirectoryChooser();  
    final File directory = chooser.showDialog(null);  
    if (directory != null) {  
        directoryTF.setText(directory.getAbsolutePath() + "\\");  
    }  
}
```

Where directoryTF is the TextField under the Label "Project Directory". Pressing the cancel button results in the cancelButtonAction method being executed in the ProjectController class.

```
private void cancelButtonAction(ActionEvent event) {
    controller.closePopup();
}
```

Which invokes the closePopup method in the MainWindowController class.

```
public void closePopup() {
    popupStage.close();
}
```

Where popupStage is a JavaFX Stage or window on which all pop up content is displayed for the application. Pressing the accept button results in the acceptButtonAction method being executed in the ProjectController class

```
@FXML
private void acceptButtonAction(ActionEvent event) {
    String name, dir;
    name = nameTF.getText();
    dir = directoryTF.getText();
    if (!"".equals(name) && !"".equals(dir)) {
        controller.createProject(name, dir);
    }
}
```

The method invokes the function createProject in the MainWindowController class provided both the Project name Text Field and Project Directory Text Field are not blank.

```
public void createProject(String name, String directory) {
    project = new JPLCProject(directory, name);
    ladderDiagram = project.getDefaultLD();
    ladderPane.setContent(ladderDiagram);
    closePopup();
}
```

Hence in this case, the user interaction layer transfers textual information pertaining to the project to the Project layer.

5.1.2.2 Project Layer

The JPLCProject class contained in the com.vts.jplceditor.project package uses the directory and project name to create an instance of the JPLCProject class which represents the entire project. When instantiated with these parameters, the instance of the class created is invoked through the method:

```
public JPLCProject(String path, String projectName) {
    this.path = path + projectName + "\\";
    this.projectName = projectName;
    File dir = new File(this.path);
    dir.mkdir();
    ladderDiagrams = new ArrayList<>();
    tags = new ArrayList<>();
    createDefault();
}
```

The `createDefault` method results in the creation of an instance of the project files and the `LadderDiagram` class.

```
private void createDefault() {
    ladderDiagrams.add(new LadderDiagram());
    projectFile = new File(path + projectName + ".xml");
    asmFile = new File(path + projectName + ".asm");
    hexFile = new File(path + projectName + ".hex");
    ldFile = new File(path + projectName + ".ld");
    programFile = new File(path + projectName + ".pgm");
    compiler = new LDCompiler();
}
```

The instance of the `LadderDiagram` class created is assigned to the `ladderDiagram` instance in the `MainWindowController` class at the `createProject` method and this `ladderDiagram` instance is placed on the user interface `ContentPane` via the assignments:

```
ladderDiagram = project.getDefaultLD();
ladderPane.setContent(ladderDiagram);
```

5.1.2.3 Ladder diagram Layer

The `LadderDiagram` class of the `com.vts.jplceditor.ld` package extends the JavaFX `Pane` class and when it is instantiated with no parameters, the method from the class invoked is:

```
public LadderDiagram() {
    super.setWidth(width);
    super.setHeight(height);
    sourceNodes = new ArrayList<>();
    rungs = new Group();
    nodes = new Group();
    getChildren().add(rungs);
    getChildren().add(nodes);
    rungList = new ArrayList<>();
    createRail();
    createDefaultRung();
}
```

The method assigns the width and height to the `Pane` inherited from `Pane` class then creates instances of the following Ladder Diagram components:

- `sourceNodes` `ArrayList`. Which is a list of all the source nodes from where all rungs will branch from.
- `rungs` `Group`. Which is a JavaFX `Group` for grouping all the Rungs which will be displayed on the ladder diagram.
- `nodes` `Group`. Which groups all the other nodes of the ladder diagram.
- `rungList` `ArrayList`. Which is a list containing information about all the rungs which will be created.

The `createRail` method in the `LadderDiagram` class creates the Rail for the ladder diagram and attaches it to the `LadderDiagram`.

```
private void createRail() {
    rungLayout = new RailLayout(width, height);
    sRail = new LDLine(rungLayout.getRailStart(),
rungLayout.getRailEnd());
    sRail.setStrokeWidth(4);
    getChildren().add(sRail);
}
```

RailLayout class of the package com.vts.jplceditor.ld.layout package computes the location of the source nodes on the ladder diagram and provides this information depending on the width and height of the ladder diagram. The LDLine class of the com.vts.jplceditor.ld.component package extends the JavaFX Line class and draws a line using the x and y coordinates provided from the JavaFX Point2D class required at instantiation.

```
public LDLine(Point2D startPoint, Point2D endPoint) {
    setStartX(startPoint.getX());
    setStartY(startPoint.getY());
    setEndX(endPoint.getX());
    setEndY(endPoint.getY());
    setStrokeWidth(2);
    setId(UUID.randomUUID().toString());
}
```

This draws a line using the points computed by the RailLayout class and assigns a unique ID to the rail for later referencing.

The createDefaultRung method in the LadderDiagram class then creates the default rung with two contact nodes using the first source node provided by the RailLayout class.

```
private void createDefaultRung() {
    SNode point = rungLayout.getSourceNodes().get(0);
    LDRung r = new LDRung(point, new Size(width,
RailLayout.RUNG_SIZE), 2);
    nodes.getChildren().add(r.getNode());
    rungList.add(r);
    rungs.getChildren().add(r);
    getChildren().add(r.getInstructions());
}
```

The LDRung class like the LadderDiagram class extends the Pane class and when instantiated with the Point2D source node and rung size:

```
public LDRung(SNode sNode, Size size, int iCount) {
    this.sNode = sNode;
    this.size = size;
    sPoint = sNode.getLocation();
    tPoint = sPoint.add(size.getWidth(), 0);
    cornerPoint = sPoint.subtract(0, size.getHeight() / 2);
    setPrefWidth(size.getWidth());
    setPrefHeight(size.getHeight());
    rungLines = new LDLineManager();
    instructions = new LDInstructionManager(sPoint);
    branches = new Group();
    instructionGroup = new Group();
```

```

instructionGroup.getChildren().add(instructions);
listeners = new ArrayList<>();
branchList = new ArrayList<>();
getChildren().add(branches);
createDefaultRung(iCount);
buildMenu();
}

```

The SNode class of the com.vts.jplceditor.ld.component.node package represents a source node with a specific location and information about the leaves branching from this source node. The SNode class extends the Vertex class, which extends the JavaFX Circle class. It implements the LDNode class meaning it shares the following common information with the other nodes, specified in the LDNode interface:

```

public interface LDNode {
    public void setInDegree(int inDegree);
    public void setOutDegree(int outDegree);
    public int getInDegree();
    public int getOutDegree();
    public NodeType getNodeType();
}

```

The LDRung rung class then uses the location of the source node to calculate the expected position of the termination point, name tPoint which also represents the termination point of the rung. It then creates an instance of the following classes:

- LineManager class which will handle the drawing of all the LDLines on the ladder diagram and contain a list of all the lines.
- LDInstructionManager class which handles the layout of all contact nodes and updates their positions every time a new contact node is added. These will be explained in detail later on.

The LDRung class then uses the positions computed by the LDInstructionManager class to position the two default contact nodes. The createDefaultRung method creates and positions the contact nodes specified by the number iCount, and attaches them to its list of children.

```

private void createDefaultRung(int iCount) {
    getChildren().add(rungLines);
    //getChildren().add(instructions);
    if (iCount == 2) {
        double dist = tPoint.getX() - sPoint.getX();
        LDInstruction i1 = new LDInstruction(sPoint.add(dist * 0.25f, 0));
        LDInstruction i2 = new LDInstruction(sPoint.add(dist * 0.75f, 0));
        i1.registerChangeListener(this);
        i2.registerChangeListener(this);
        if (position != null) {
            i1.setPosition(new LDPosition((short) 0,
position.getyPosition()));
            i2.setPosition(new LDPosition((short) 1,
position.getyPosition()));
        } else {
            i1.setPosition(new LDPosition((short) 0, (short) 0));
            i2.setPosition(new LDPosition((short) 1, (short) 0));
        }
        instructions.addInstruction(0, i1);
        instructions.addInstruction(1, i2);
    }
}

```

```
sNode.addChild(i1);
sNode.addChild(i2);

rungLines.addLine(sPoint, i1.getEndPoint());
rungLines.addLine(i1.getEndPoint(), i2.getEndPoint());
rungLines.addLine(i2.getEndPoint(), tPoint);
} else if (iCount == 1) {
    double dist = tPoint.getX() - sPoint.getX();
    LDInstruction il = new LDInstruction(sPoint.add(dist * 0.33f,
0));
    il.registerChangeListener(this);
    instructions.setMinInstructionCount((short) iCount);
    instructions.addInstruction(0, il);
    sNode.addChild(il);

    rungLines.addLine(sPoint, il.getEndPoint());
    rungLines.addLine(il.getEndPoint(), tPoint);
}
}
```

5.1.3 Creating a tag

A tag is created by pressing the create tag menu item from the Project menu. This displays the Create tag window shown in figure 5.7 below.

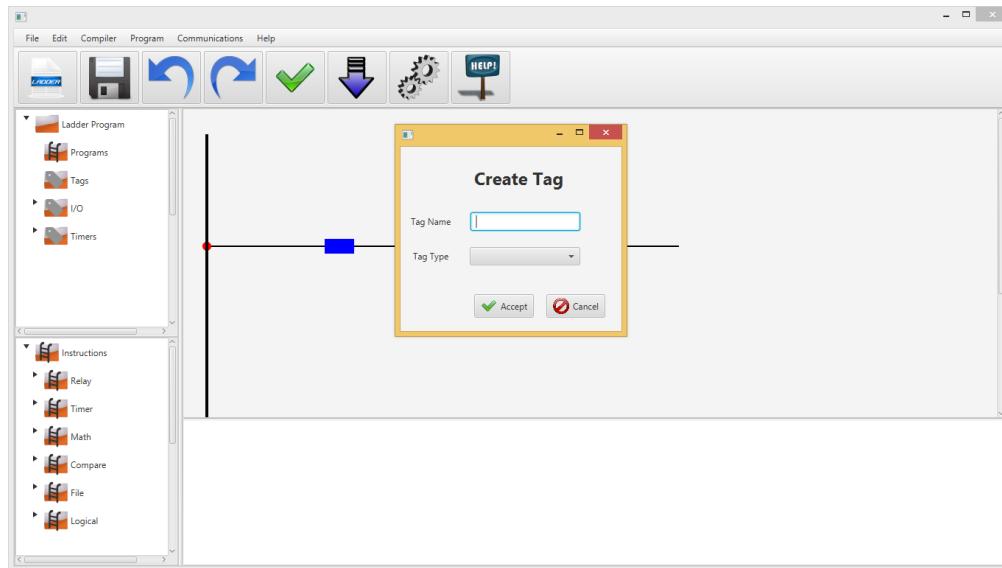


Figure 5.7 Create Tag pop up window

After entering the tag name and selecting the tag type from the combo box, the tag is displayed under the program files in the tags folder.

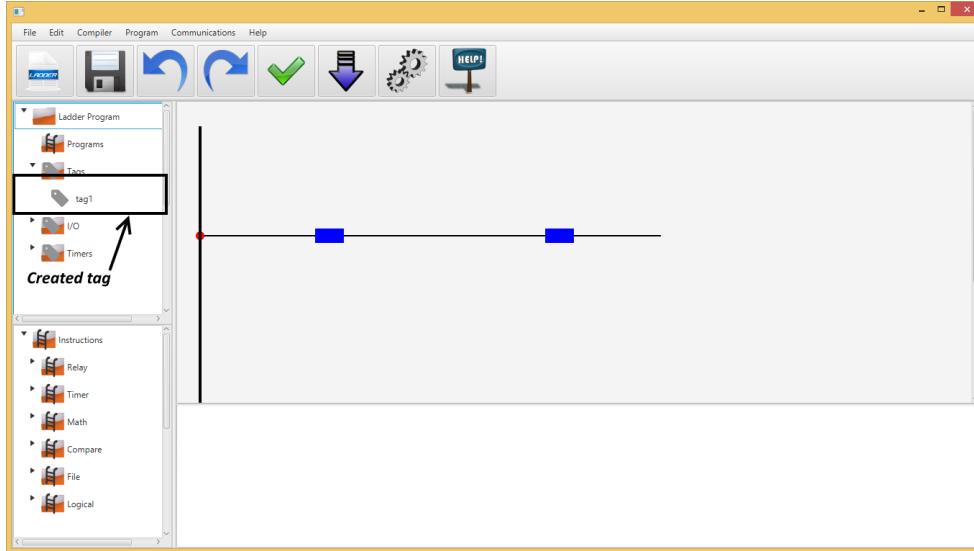


Figure 5.8 Created tag displayed under the tags folder

5.1.3.1 Implementation in code

Creating a tag begins at the User Interaction layer when the user selects the Create Tag option from the menu bar. This invokes the `createTag` method in the `MainWindowController` class.

```
@FXML  
private void createTag(ActionEvent event) {  
    url = getClass().getResource(R.WINDOW_TAG);  
    showPopup(Popup.TagWindow);  
}
```

The `showPopup` method displays the Create Tag window whose FXML controller is the `TagWindowController` of the `com.vts.jplceditor.ui.controller` package. After the user enters the tag name and specifies the tag type then presses the accept button, the `acceptButtonAction` method of the `TagWindowController` class is executed.

```
@FXML  
private void acceptButtonAction(ActionEvent event) {  
    String name = nameTF.getText();  
    String type = typeCB.getValue();  
    if(name != "" & type != ""){  
        controller.addTag(name, type);  
    }  
}
```

This passes these parameters to the `addTag` method in the `MainWindowController` class.

```
public void addTag(String tagName, String tagType) {  
    ImageView tagIcon = new ImageView(new Image(R.ICON_TAG_ICON));  
    tagList.getChildren().add(new TreeItem<>(tagName, tagIcon));  
    tags.put(tagName, tagType);  
    PLCDataType type = UIUtil.typeFromString(tagType);  
    project.addTag(type, tagName);
```

```
        closePopup();  
    }  
}
```

The method does two things. First it obtains an ImageView to display the tag on the tag tree view of the GUI. Then it stores the tag information in the JPLCProject object for the class.

5.1.4 Creating a Rung

A rung is created when the user presses the Create Rung menu item from the Project menu bar. This displays the create Rung window shown below in figure 5.9.

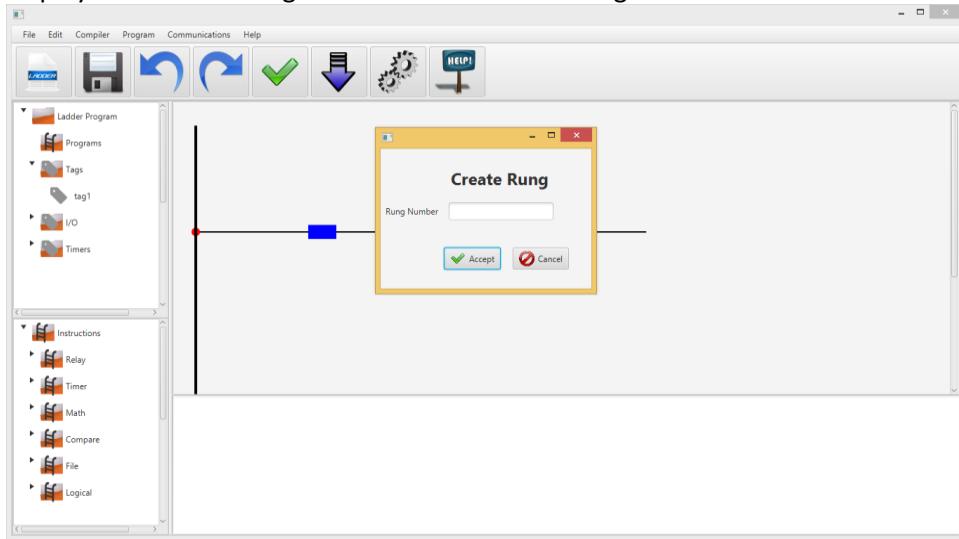


Figure 5.10 Create Rung pop up window

After the user enters the rung number with zero indicating the first position, the rung is added to the Ladder Diagram.

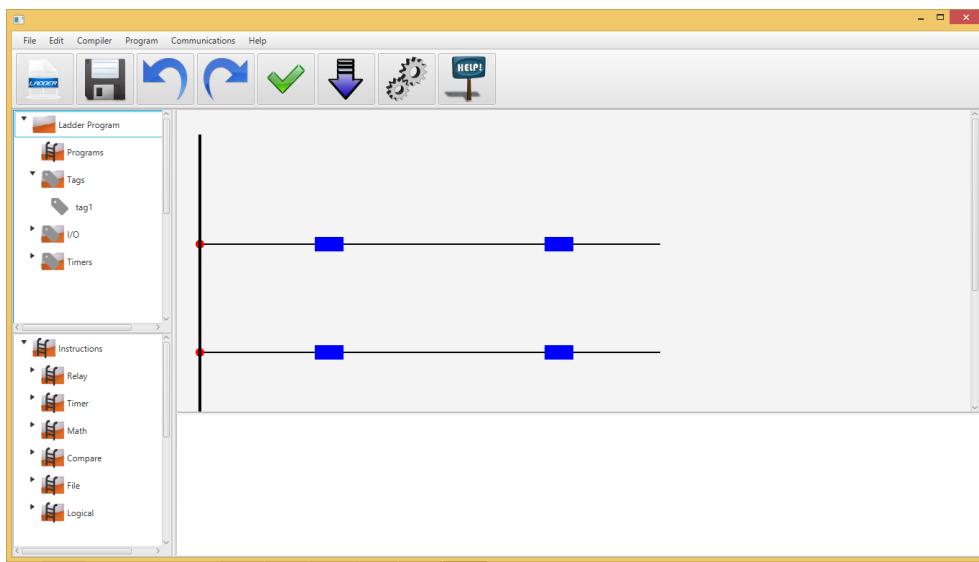


Figure 5.11 New Rung added to Ladder Diagram

5.1.4.1 Implementation in code

Creating a rung begins at the User Interaction layer when the user selects the Create Rung option from the menu bar. This invokes the `createRung` method in the `MainWindowController` class.

```
@FXML  
private void createRung(ActionEvent event) {  
    url = getClass().getResource(R.WINDOW_RUNG);  
    showPopup(Popup.RungWindow);  
}
```

The method displays the Create Rung window whose JavaFX controller is the `RungViewController` class of the `com.vts.jplceditor.ui.controller` class. After the user enters the rung number and presses the accept button, the `acceptButtonAction` of the `RungViewController` class is invoked.

```
@FXML  
private void acceptButtonAction(ActionEvent event) {  
    if (isInteger(numberTF.getText())) {  
        int number = Integer.valueOf(numberTF.getText());  
        controller.addRung(number);  
    }  
}
```

The method invokes the `MainWindowController` method `addRung` with the parameter being the number the user entered.

```
public void addRung(int rungNumber) {  
    ladderDiagram.addRung(rungNumber);  
    closePopup();  
}
```

The method then adds the rung to the `ladderDiagram` object. The `addRung` method shown below is from the `LadderDiagram` class and adds the rung to the specified index, then the entire Ladder diagram is redrawn with the second rung.

```
public void addRung(int position) {  
    rungLayout.addNewSourceNode();  
    List<SNode> sNodes = rungLayout.getSourceNodes();  
    int count = sNodes.size();  
    SNode sNode = rungLayout.getSourceNodes().get(count - 1);  
    LDRung r = new LDRung(sNode, new Size(width, RailLayout.RUNG_SIZE), 2);  
    nodes.getChildren().add(sNode);  
    rungList.add(position, r);  
    rungs.getChildren().add(r);  
    getChildren().add(r.getInstructionGroup());  
  
    for (int i=0;i<count;i++) {  
        rungList.get(i).relocateToPoint(sNodes.get(i).getLocation());  
    }  
}
```

5.1.5 Dragging data onto the Ladder Diagram

Instructions or data can be dragged from the side bar onto the ladder diagram instruction nodes (represented by the blue boxes) or the instruction data fields. Whenever the user drags data over an instruction node, it changes color to indicate that it can accept dragged data. When the user drags the data away from the contact node, it returns to its original color.

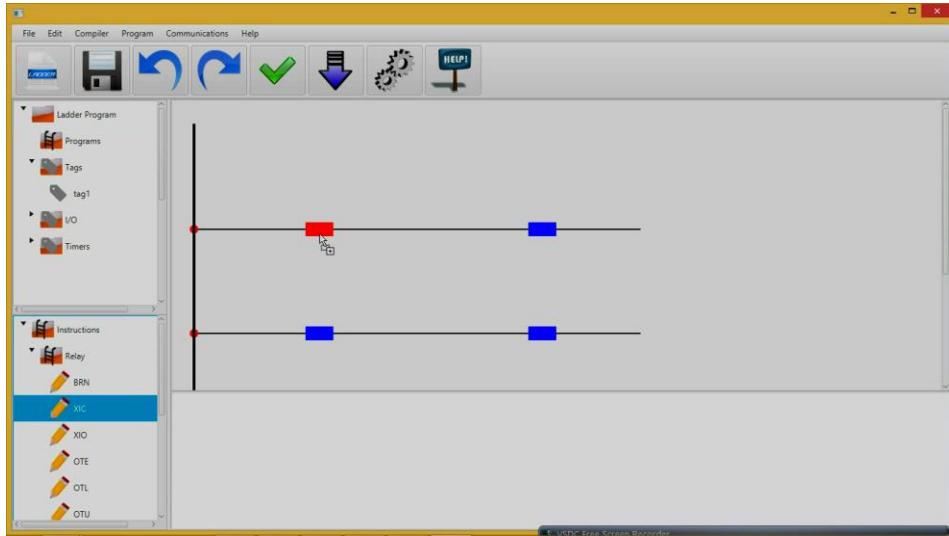


Figure 5.1.1 User dragging XIC instruction onto an instruction node

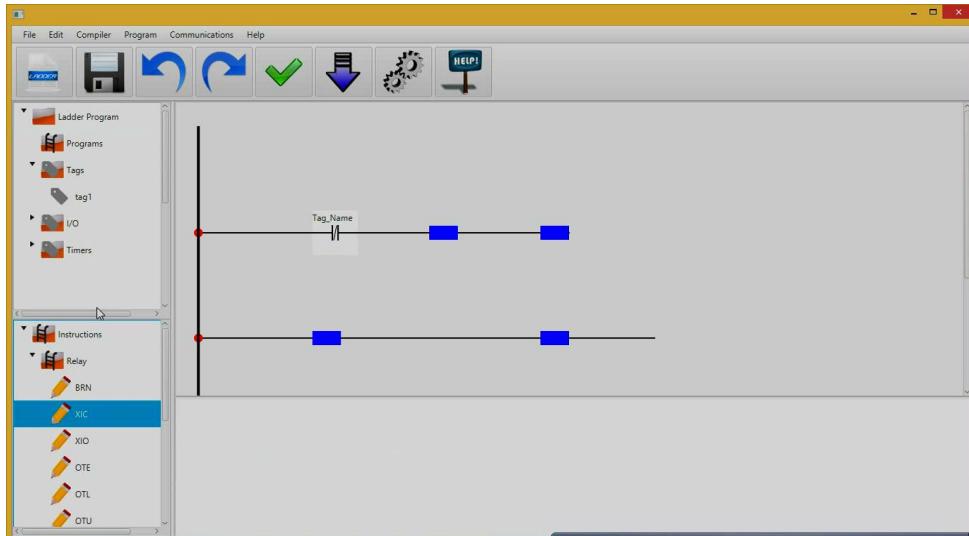


Figure 5.1.2 XIC instruction dropped on an instruction node

After the user drops the instruction, it appears on the instruction node and a new blank instruction node is positioned after it. This is to allow the user to add more instructions onto the rung and by default, each rung can have a minimum of two instruction nodes.

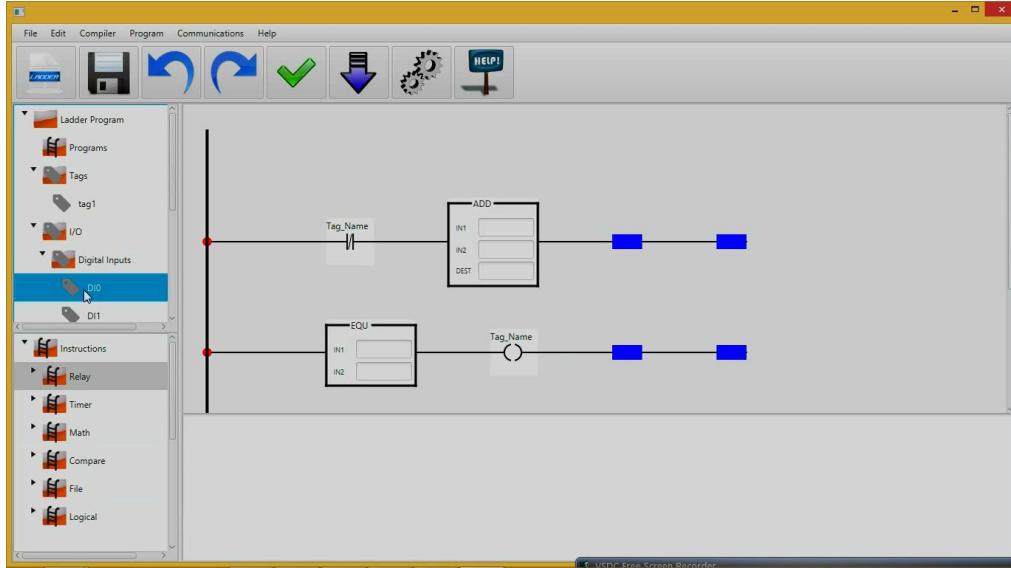


Figure 5.1.3 More instructions dropped onto Ladder Diagram

Once instructions have been added onto the ladder diagram, the next thing is to specify the parameters on them. JPLC Editor allows the user to drag tags onto instructions, or type in the name of the instruction on function blocks.

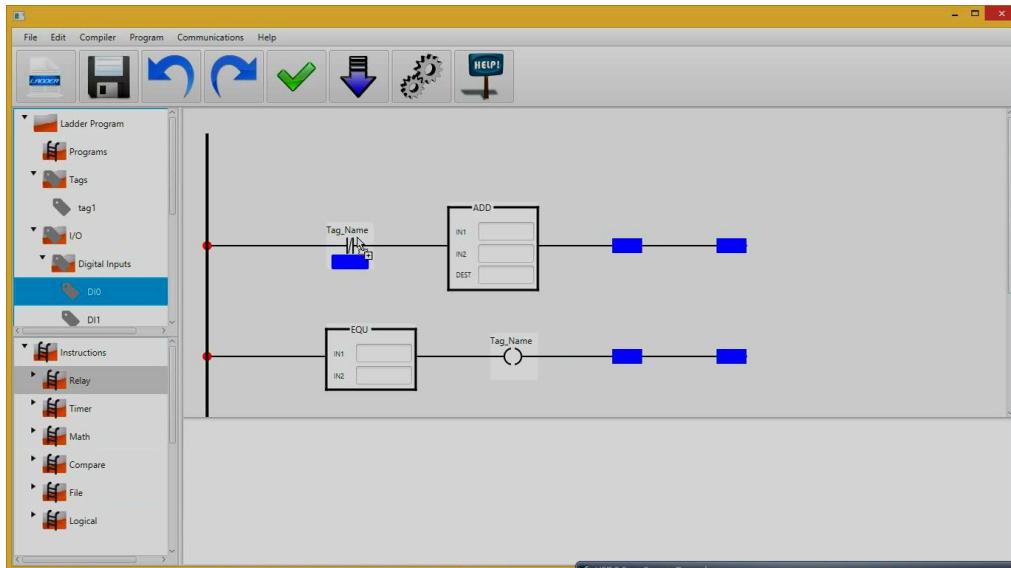


Figure 5.1.4 User dragging a tag onto an instruction

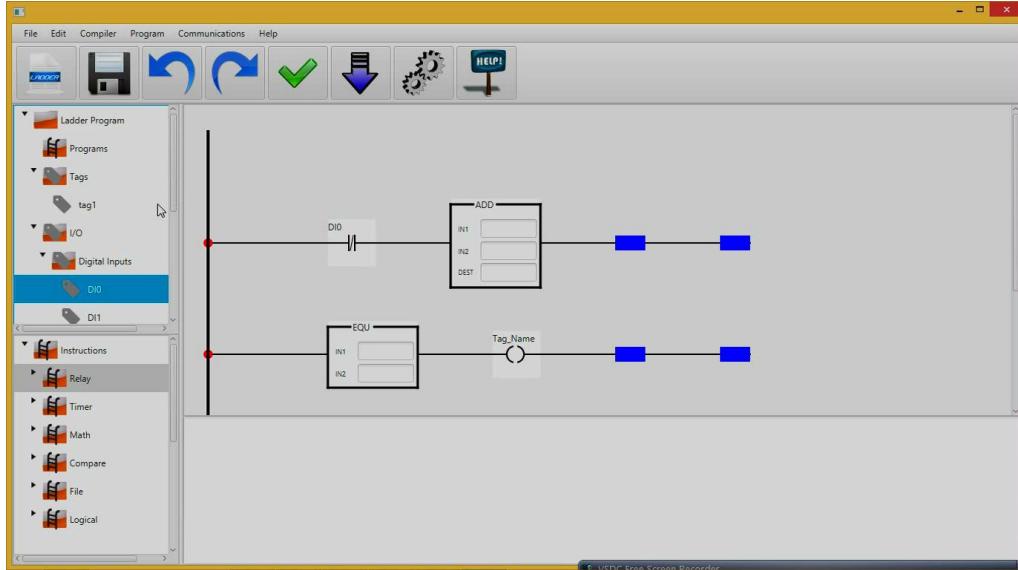


Figure 5.1.5 Digital Input 0 tag dropped onto XIC instruction

JPLC Editor provides the tags for all Inputs and outputs then allows the user create internal tags which are to be stored in RAM.

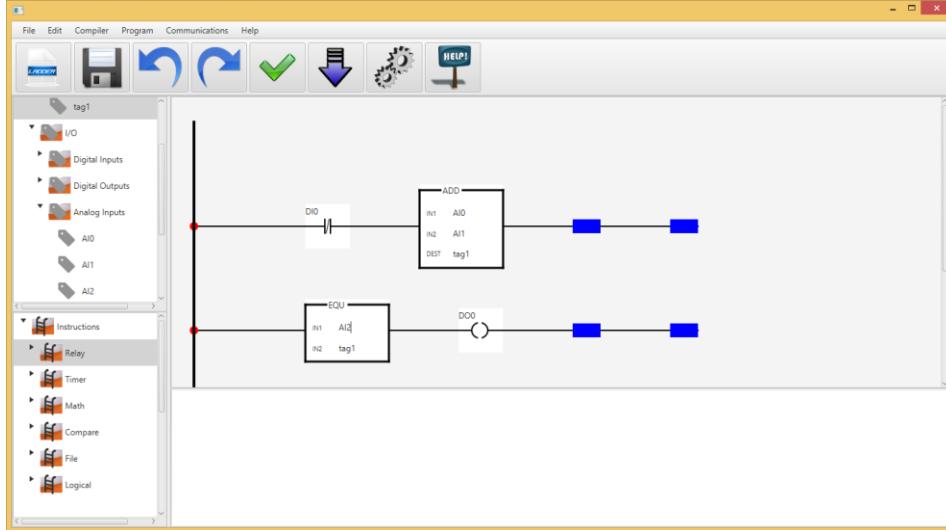


Figure 5.1.6 More tags dropped onto other instructions

5.1.5.1 Implementation in code

Dragging of data begins at the User Interaction Layer when the users drags data from either tree view on the side bars.

On creation of the MainWindow FXML file, both JavaFX TreeView objects are given “onDragDetected” events.

```
<TreeView fx:id="programTree" minHeight="600.0" minWidth="200.0"
onDragDetected="#onProgramDragDetected" />
.
.
```

```
<TreeView fx:id="instructionTree" minHeight="600.0" minWidth="200.0"
onDragDetected="#onInstructionDragDetected" />
```

Which means when the upper Tree View separated by the scroll bar of figure 5.6 detects a drag, the onProgramDragDetected method of the MainWindowController class is invoked. When the bottom Tree View detects a drag, the OnInstructionDragDetected method is invoked.

5.1.5.2 Dragging instructions onto Instruction nodes

The onInstructionDragDetected method is invoked whenever the user drags an instruction.

```
private void onInstructionDragDetected(MouseEvent event) {

    String selected =
instructionTree.getSelectionModel().getSelectedItem().getValue();
    String parent =
instructionTree.getSelectionModel().getSelectedItem().getParent().getValue();
    Instruction i = InstructionUtil.getInstruction(selected);
```

The method first identifies name and parent of the selected instruction. The parent in this case being the instruction group either Relay, Math and so on, as displayed on the tree view.

```
if (i != null) {
    dragBoard = instructionTree.startDragAndDrop(TransferMode.COPY);
    dragContent = new ClipboardContent();
    dragContent.putString(selected);
    dragBoard.setContent(dragContent);
} else {
    System.out.println("Null Instruction:" + selected);
}
event.consume();}
```

Then it checks if the dragged item has a parent to prevent the user from dragging folders. The method then starts a drag and drop event from the Tree View putting the name of the dragged instruction on the Clip board.

The LDInstruction class of the com.vts.jplceditor.ld.component package represents the instruction node on which the instruction can be dropped onto. It is designed to respond to four drag events:

- i. DragEntered event, when user data into its region drags.
- ii. DragExited event, when the user drags data away from its region.
- iii. DragOver event, when user data over its region drags
- iv. DragDropped event, when the user drops data onto its region.

It appears as a blue box on the ladder diagram and changes its background colour to red when the user drags data into it then returns to the blue colour when the user drags data away from it.

```
public LDInstruction(Point2D startPoint) {
    this.startPoint = startPoint;
    //Creates the default blue color of the square block
    fill = new BackgroundFill(Color.BLUE, CornerRadii.EMPTY, Insets.EMPTY);
    fillDragOver = new BackgroundFill(Color.RED, CornerRadii.EMPTY, Insets.EMPTY);
    fillDragDropped =new BackgroundFill(Color.WHITE, CornerRadii.EMPTY,
    Insets.EMPTY);
```

```

        Background background = new Background(fill);
        setBackground(background);
        endPoint = startPoint.add(size.getWidth(), 0);
        cornerPoint = startPoint.subtract(0, size.getHeight() / 2);
        setLayoutX(cornerPoint.getX());
        setLayoutY(cornerPoint.getY());
        setPrefWidth(size.getWidth());
        setPrefHeight(size.getHeight());
        //Assigns a unique ID to this Node
        setId(UUID.randomUUID().toString());
        listeners = new ArrayList<>();
        createDragHandlers();
    }

}

```

The Drag events mentioned earlier are created in the createDragHandlers method which is in the same class. When the user drops data onto its region, the drag event that responds to the drop event is:

```

dragDropped = new EventHandler<DragEvent>() {
    public void handle(DragEvent event) {
        Dragboard db = event.getDragboard();
        if (db.hasString()) {
            String name = db.getString();

```

The method first gets the string object contained in the Drag Event Drag board. The method then checks if the dragged item name is a valid instruction name by passing the name of the item to the fromID method of the PLCOperatorDecoder class. The method which will return null if it was an invalid instruction name or the the PLCOperator class object created from a valid instruction name.

```

        PLCOperator i = PLCOperatorDecoder.fromID(name);
        if (i != null) {
            setBackground(new Background(fillDragDropped));
            setInstruction(i);
        }
        event.setDropCompleted(true);
        event.consume();
    }
};

}
;

```

When the name is valid, the setInstruction method of the LDInstruction class is invoked.

```

public void setInstruction(PLCOperator operator) {
    if (!getChildren().isEmpty()) {
        getChildren().remove(0);
    }
    if (operator != PLCOperator.NULL) {
        instruction = InstructionUtil.getInstruction(operator.name());
        AnchorPane node = instruction.getInstance();
        size.setWidth(node.getPrefWidth());
        size.setHeight(node.getPrefHeight());
        setPrefWidth(size.getWidth());
        setPrefHeight(size.getHeight());
    }
}

```

```
        getChildren().add(0, node);
        relocateToPoint(startPoint);
        hasInstruction = true;
        if (listeners.size() > 0) {
            for (int i = 0; i < listeners.size(); i++) {
                listeners.get(i).onInstructionAdded(getId());
            }
        }
    }
}
```

The function first checks if the contact node had an instruction then removes it. It then uses the PLCOperator object to obtain the “user interface” instruction obtained from the getInstruction method of the InstructionUtil class. The Instruction Interface of the package com.vts.jplceditor.id.instruction represents a Java object with an Anchor Pane containing the “user interface” instruction as well the “compilation instruction” which contains information needed to compile the ladder diagram.

```
    public interface Instruction {  
        public AnchorPane getInstance();  
        public PLCInstruction getPLCInstruction();  
    }
```

The anchorpane represents a custom JavaFX component created using the JavaFX Scene builder application to represent a PLC instruction. All of the “user interface” instructions are in the package com.vts.jplceditor.lid.instruction package separated by the instruction category.

5.1.5.3 Dragging data onto Instructions

The `onProgramDragDetected` method also gets the selected item and its parent but this time from the Program Tree view. After verifying the dragged data is a tag, the information is formatted to specify the tag type before putting it onto the drag board. The format is of the form:

Tag tagName tagType

Where tag identifies the data as a tag.

```
private void onProgramDragDetected(MouseEvent event) {
    dragBoard = programTree.startDragAndDrop(TransferMode.COPY);
    dragContent = new ClipboardContent();
    String selected =
programTree.getSelectionModel().getSelectedItem().getValue();
    String parent =
programTree.getSelectionModel().getSelectedItem().getParent().getValue();

        boolean isTag = parent.equals("Tags") || parent.equals("Digital
Inputs")
                || parent.equals("Digital Outputs") || parent.equals("Analog
Inputs") || parent.equals("Analog Outputs");
        if (isTag) {
            String type = tags.get(selected);
            String dragData = DragDataFormatter.formatTag(selected, type);
            dragContent.putString(dragData);
            dragBoard.setContent(dragContent);
```

```

        }
        if (parent.equals("Timer") || parent.equals("Counter")) {
            String type;
            if (selected.contains("TT") || selected.contains("DN")) {
                type = DragDataFormatter.TagType_BIT;
            } else {
                type = DragDataFormatter.TagType_INT;
            }
            String dragData = DragDataFormatter.formatTag(selected, type);
            dragContent.putString(dragData);
            dragBoard.setContent(dragContent);
        }

        event.consume();
    }
}

```

The user interface instructions each with their own JavaFX controllers respond to drag dropped events and set parameters within the instruction depending on the dragged data. An example of such an instruction is the Ote class of the package com.vts.jplceditor.ld.instruction.relay package whose dragDropped event is inside the buildHandlers method.

```

dragDropped = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureTarget() == this) {
            entered = false;
        }
        Dragboard db = event.getDragboard();
        boolean success = false;
    }
}

```

The method then gets the formatted dragged data and decodes to see if it is a tag and if the data type is the one required by the instruction. In this case, the expected data type is the bit data type.

```

if (db.hasString()) {
    String data = db.getString();
    String dataType = DragDataFormatter.getTagType(data);
    if (dataType.equals(DragDataFormatter.TagType_BIT)) {
        String name = DragDataFormatter.getTagName(data);
        tagName.setText(name);
    }
}
event.setDropCompleted(success);
event.consume();
}
;
}

```

5.1.6 Saving and Opening a Project

A project is saved when the user presses the save button on the quick launch menu or presses the Save project menu item from the File menu bar.

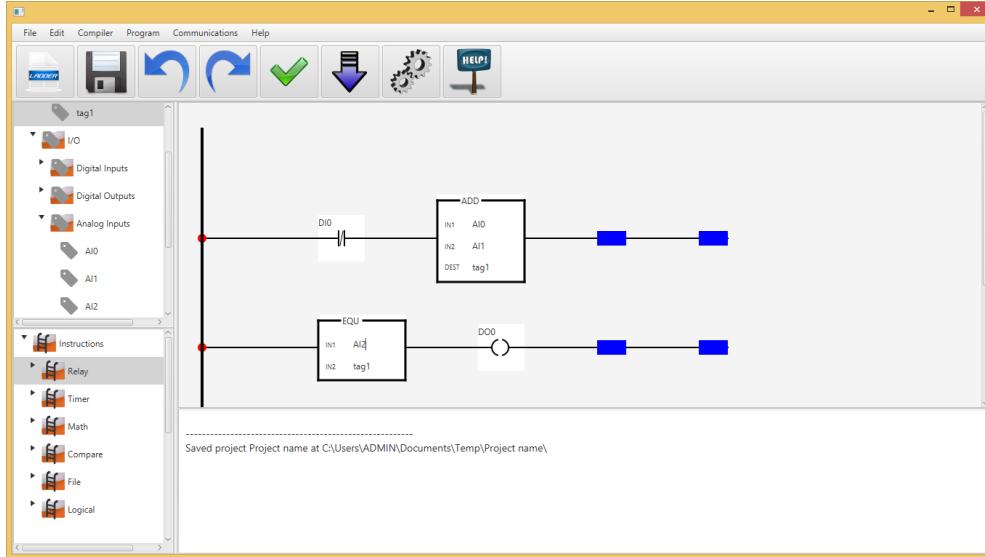


Figure 5.1.7 Saving of a JPLC Editor project with status indicated

The status field then displays the status of the save attempt. When successfully saved, the project file is created.

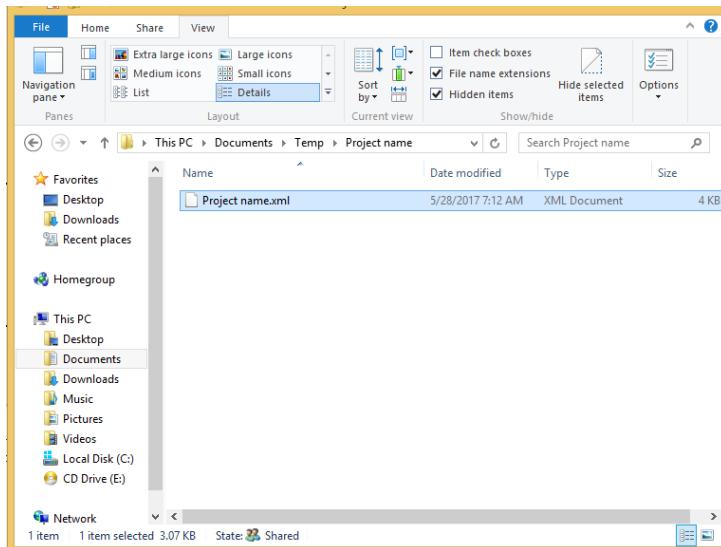


Figure 5.1.8 Project xml file saved in working directory

The xml contents of the file are shown by the code below

```
<?xml version="1.0" encoding="UTF-8"?>
<Project>
<directory>C:\Users\ADMIN\Documents\Temp\Project name\</directory>
<projectTitle>Project name</projectTitle>
<ladderDiagramList>
  <LadderDiagram>
    <rungs>
      <Rung>
        <size>
          <width>640.0</width>
```

```
    <height>150.0</height>
  </size>
  <sNode>
    <nodeType>Source</nodeType>
    <locationX>32.0</locationX>
    <locationY>186.0</locationY>
  </sNode>
  <instructions>
    <Instruction>
      <locationX>192.0</locationX>
      <locationY>186.0</locationY>
      <size>
        <width>64.0</width>
        <height>64.0</height>
      </size>
    </Instruction>
    <Instruction>
      <locationX>356.0</locationX>
      <locationY>186.0</locationY>
      <size>
        <width>120.0</width>
        <height>120.0</height>
      </size>
    </Instruction>
    <Instruction>
      <locationX>576.0</locationX>
      <locationY>186.0</locationY>
      <size>
        <width>40.0</width>
        <height>20.0</height>
      </size>
    </Instruction>
    <Instruction>
      <locationX>716.0</locationX>
      <locationY>186.0</locationY>
      <size>
        <width>40.0</width>
        <height>20.0</height>
      </size>
    </Instruction>
  </instructions>
</Rung>
<Rung>
  <size>
    <width>640.0</width>
    <height>150.0</height>
  </size>
  <sNode>
    <nodeType>Source</nodeType>
    <locationX>32.0</locationX>
    <locationY>336.0</locationY>
  </sNode>
  <instructions>
    <Instruction>
      <locationX>192.0</locationX>
      <locationY>336.0</locationY>
      <size>
```

```

        <width>120.0</width>
        <height>90.0</height>
    </size>
</Instruction>
<Instruction>
    <locationX>412.0</locationX>
    <locationY>336.0</locationY>
    <size>
        <width>64.0</width>
        <height>64.0</height>
    </size>
</Instruction>
<Instruction>
    <locationX>576.0</locationX>
    <locationY>336.0</locationY>
    <size>
        <width>40.0</width>
        <height>20.0</height>
    </size>
</Instruction>
<Instruction>
    <locationX>716.0</locationX>
    <locationY>336.0</locationY>
    <size>
        <width>40.0</width>
        <height>20.0</height>
    </size>
</Instruction>
</instructions>
</Rung>
</rungs>
<sNodes/>
</LadderDiagram>
</ladderDiagramList>
<tagList>
    <Tag>
        <dataType>BYTE</dataType>
        <identity>tag1</identity>
    </Tag>
</tagList>
</Project>
```

5.1.6.1 Implementation in code

The project information is saved as an xml file. Saving a project begins at the User Interaction layer when the user clicks the save button. The `saveProject` method is invoked inside the `MainWindowController` class

```
@FXML
private void saveProject(ActionEvent event) {
    if (project != null) {
        try {
            status.appendText(project.saveProject().toString());
        } catch (IOException ex) {
```

```
        Logger.getLogger(MainWindowController.class.getName()).log(Level.SEVERE,  
        null, ex);  
    }  
}  
}
```

Where status is the status field on the main GUI and appends the text returned by the saveProject method of the JPLCProject class.

```
public String saveProject() throws IOException {
    StringBuilder status = new StringBuilder();
    status.append("\n");
    XStream xs = XmlAlias.projectAlias();
    List<IOLD> ldList = fromLadderDiagramList();
    SavableProject p = new SavableProject();
    p.setDirectory(path);
    p.setLadderDiagramList(ldList);
    p.setProjectTitle(projectName);
    p.setTagList(tags);
    String content = xs.toXML(p);
    XmlFile.writeFile(content, projectFile);
    status.append("-----");
    status.append("\n");
    status.append("Saved project ").append(projectName).append(" at ");
    status.append(path);
    status.append("\n");
    return status;
}
```

Saving the project involves parsing the Ladder diagram information into a model tree for the ladder diagram. The model for the LadderDiagram class being the IOLD class, for the LDRung class being the IORung class, for the SNode class being the IONode class and for the LDInstruction class being the IOInstruction class. Each model stores information required to reproduce the Ladder diagram.

Once the model for the ladder diagram is produced, this model is saved into an xml file using the XStream java library for saving xml files [17].

Opening a project is the reverse process of saving it. This happens when the user clicks the open button and the `openProject` method of the `MainWindowController` class.

```
@FXML  
private void openProject(ActionEvent event) {  
    FileChooser chooser = new FileChooser();  
    final File projFile = chooser.showOpenDialog(null);  
    if (projFile != null) {  
        SavableProject pro = JPLCProject.readInProject(projFile);  
        project = new JPLCProject(pro);  
        ladderDiagram = project.getDefaultLD();  
        ladderPane.setContent(ladderDiagram);  
    }  
}
```

After the user selects the file to open, this projFile is then passed as a parameter to the static JPLCProject class method readInProject which reads an xml file and transfers the information in it to the SavableProject class. The SavableProject class represents the model format of the JPLCEditor class. This savable project is then used to create the JPLCProject class instance in the MainWindowController class and the default LadderDiagram class created is attached to the ladder diagram Pane.

5.1.7 Compiling the Program

A program is compiled by pressing the compile button from the Compiler menu bar, after which the Ladder diagram removes the unused instruction nodes and generates the output files.

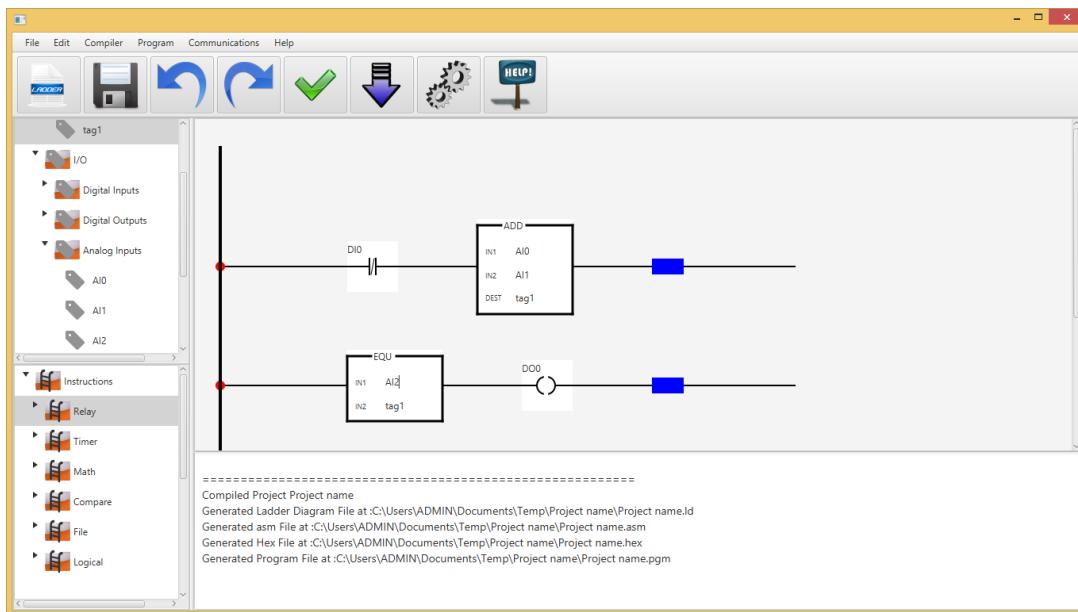


Figure 5.1.9 Compiled program with indicated status field

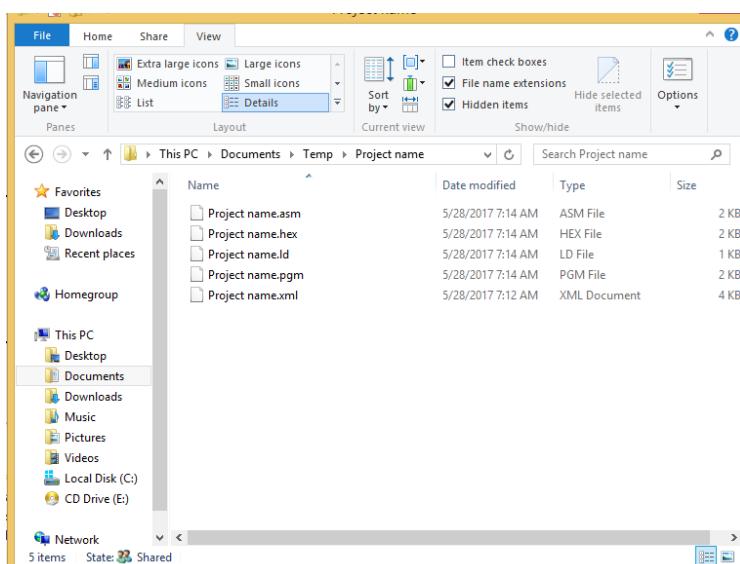


Figure 5.2.0 Compilation generated files

The contents of each file are shown in Appendix K. The compilation of the ladder diagram occurs layer by layer.

5.1.7.1 User Interaction Layer

The compilation of the program begins at the User interaction layer when the user presses compile from the Compiler menu bar. This invokes the compileProgram method of the MainWindowController class.

```
private void compileProgram(ActionEvent event) {
    if (project != null) {
        status.clear();
        status.appendText(project.compileProject().toString());
    } else {
        status.appendText("\nYou need to create a Project First\n!!!!!!\n");
    }
}
```

This then invokes the compileProject method from the JPLCEditor class which then returns a StringBuilder object indicating the status of the compilation.

```
public StringBuilder compileProject() {
    StringBuilder status = new StringBuilder();
    // if (!compiled) {
    if (compiler.compile(ladderDiagrams.get(0), tags)) {
....
```

The JPLCProject uses the LDCompiler object identified by the “compiler” identifier it has to compile the ladder diagram. After the compilation is successful, the project generates the project files.

5.1.7.2 Compiler Layer

At the compiler layer, the LDCompiler class begins the compilation of the Ladder Diagram.

```
public boolean compile(LadderDiagram ladderDiagram, List<PLCOperand> tagList)
{
    boolean success;
    ld = ladderDiagram;
    tags = tagList;
    if(!visitor.visitLD(ld)){
        success=false;
    }else{
        picCompiler.compileInstructionList(visitor.getRungList(), tagList);
        success=true;
    }
    return success;
}
```

The LDVisitor class produces a rung list which the LDCompiler class then passes to the Pic18Compiler class to generate the assembly code.

```
public void compileInstructionList(List<PLC_Rung> list, List<PLCOperand>
tagList) {
    plcInstructions = new ArrayList<>();
    picInstructions = new ArrayList<>();
```

```

        verifyTypes(list, tagList); //assign data types to instruction
operands
        assignAddresses(list, tagList); //assign addresses to operands and
tags
        //PLC_Rung
        compilePLC(list); //generate a PLC instruction list
        compilePICasm(); //generate a Pic instruction list
        compilePICHex(); // generate hex code
    }
}

```

The Pic18Compiler class then assigns addresses to all the tags to determine where the program RAM memory usage will begin. The compilePLC method uses the instructions in each rung to determine which one is the first instruction on the rung. The following then are assigned their previous instruction's enable output bit which it needs to monitor to determine whether it will execute it's code.

5.1.8 Downloading the program

The program is downloaded to the PLC emulator when the user presses the download menu item from the quick launch menu, or presses the download menu item from the Communications menu.

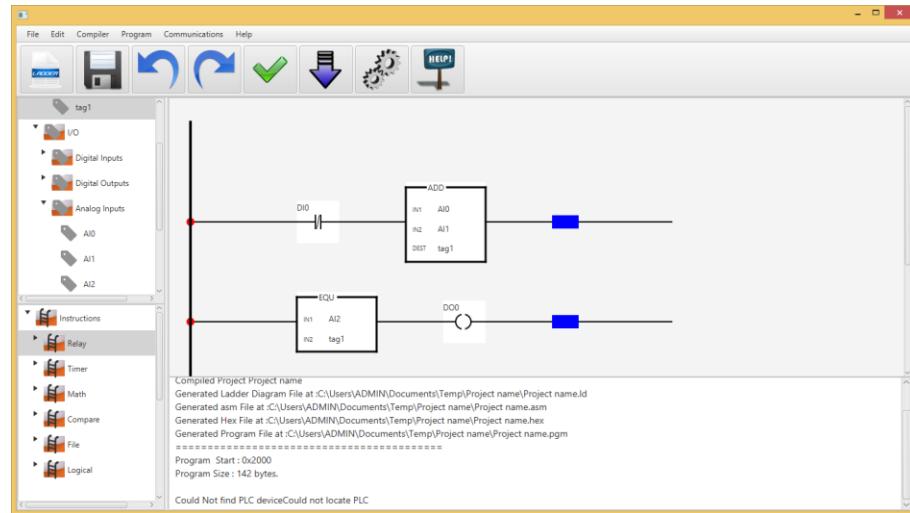


Figure 5.2.1 Application failing to locate PLC after user pressed the download button.

The application first looks to find if the program has been compiled, then looks to locate the USB port connecting the PLC emulator. When it is not located, the application stops the download process.

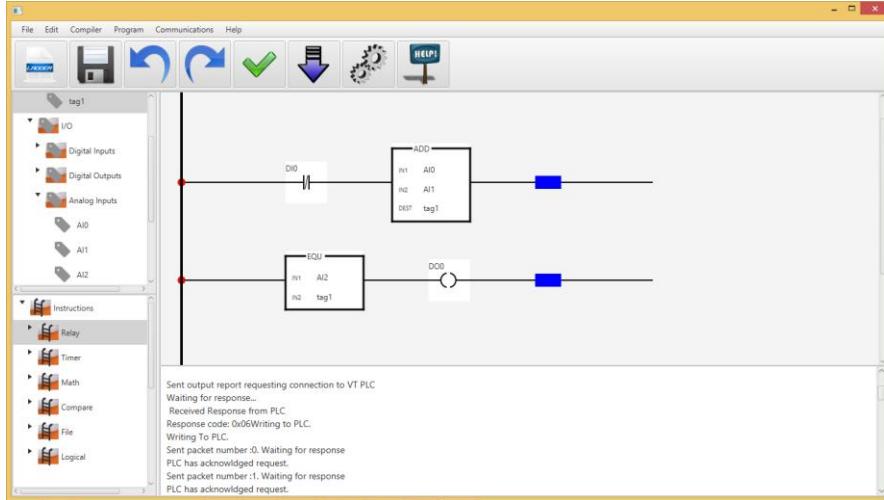


Figure 5.2.2 Application in the process of downloading the user program into the PLC

5.1.8.1 Implementation in code

When the user presses the download button, the method of the MainWindowController class is executed.

```
@FXML
private void downloadProgram(ActionEvent event) {
    if (project != null) {
        status.appendText(project.download().toString());
    } else {
        status.appendText("\nYou need to create a Project and Compile it First!!!!!!\n");
    }
}
```

This invokes the JPLCProject class's download method which obtains the program byte code from the compiled program and then sends it to the SessionManager class which handles all the JPLCProtocol formatting and transporting of the program to the PLC emulator.

```
public StringBuilder download() {
    StringBuilder status = new StringBuilder();
    status.append("\n");
    if (compiled) {
        plcSession = new SessionManager();
        Pic18Prog program = compiler.getPicProgram();
        status.append(program.getProgramInformation());
    }
    status.append(plcSession.writeToPlc(program.getCodeHex().array()));
} else {
    status.append("=====      ERROR !!!!!      ======");
    status.append("\n");
    status.append("Project ").append(this.projectName).append(" is not compiled yet !!!!");
    status.append("\n");
}
return status;
}
```

}

5.2 Processor board implementation and Testing

This section looks at the tests and simulations performed on the Processor board before it was transferred onto a prototyping board.

5.2.1 Run Mode Test simulation

Aim

To investigate whether the Main microcontroller is able to read inputs and write outputs to the I/O interface microcontroller.

Method

1. Connect the two microcontrollers as designed in figure 4.1.
2. Implement the code for the I/O interface microcontroller.
3. Implement only the code for running the user program (shown in Appendix C) with the following modifications:
 - a. Display analogue input 0 on the lcd.
 - b. Assign value of analogue input 0 to analogue output 0.
 - c. Assign the value of the digital inputs to that of the digital outputs.
4. Connect LEDs to the digital outs and switches to the digital outputs.
5. Connect an analogue input to analogue input 0 and a voltmeter to analogue output 0.

Results and analysis

As seen from the simulation in figure 5.2.3, the LED lights correspond to the digital switch values.

The analogue input 0 value was successfully read from the I/O microcontroller and assigned to the analogue output 0 at the user program microcontroller. The AO0 value written to the I/O microcontroller is displayed on the voltmeter in figure 5.2.3

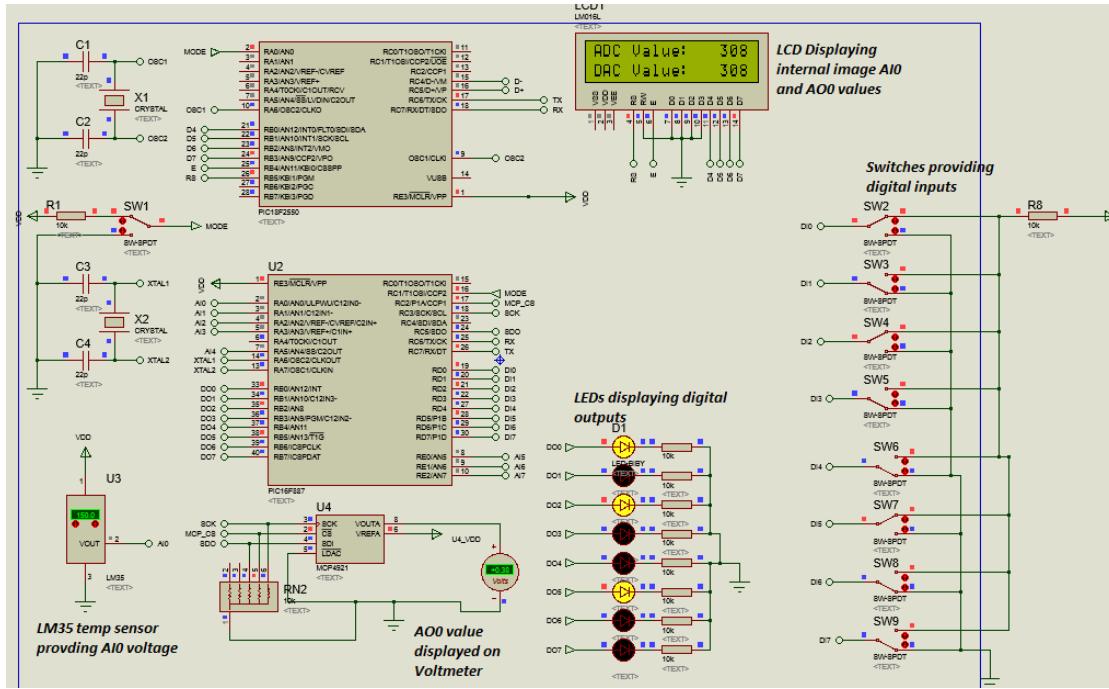


Figure 5.2.3 Proteus Run Mode simulation

The AIO value coming from the LM35 temperature sensor corresponds to a temperature of 150 degrees Celsius. The voltage that corresponds to this temperature is 1.5V and is the value sent to AIO. The ADC convertor in the I/O microcontroller has a resolution of 10 bits that is, a maximum value of 1023 corresponding to 5V. So that 1.5V would be expected to produce an ADC value of

$$N_{ADC} = \frac{1.5}{5} \times 1023 = 306$$

Which is approximately equal to the read value of 308. This value was then assigned to the AOI which went to a 12 bit DAC connected to the I/O microcontroller.

$$V_{DAC} = \frac{306}{4095} \times 5 = 0.373V$$

Which is approximately equal to the measured voltage of 0.38V on the voltmeter.

Conclusion

The user program microcontroller was able to respond to changes to all the inputs and update the outputs as expected. The digital input value was successfully read from the I/O microcontroller, then assigned to digital out value which was displayed by the LEDs.

5.2.2 Program Mode Test

Aim

To determine if the JPLC Editor desktop application is able to download the PLC program into the user program microcontroller.

Method

1. Connect the CPU board microcontroller as illustrated in figure 4.1 with the complete microcontroller programs used for both microcontrollers.
2. Develop a simple relay ladder diagram using the JPLC Editor application and compile it.
3. Run the Proteus simulation of the processor board and put the CPU in program mode.
4. Press the download button on the JPLC Editor application.
5. Record the following data:

Results and analysis

The program loaded into the emulator is shown in the figure below and it gives inconclusive evidence as to whether the program was loaded into the microcontroller. The program read from the programmed locations shows blank memory areas.

However, since the experiment was repeated at a later time with the code to first erase the PLC removed, the programming was successful but the program still did not run, hence the Proteus simulation produced inconclusive evidence as to whether the program was written to. It also raised the question of whether Proteus was capable of simulating a flash write, since it uses a static hex file for the program.

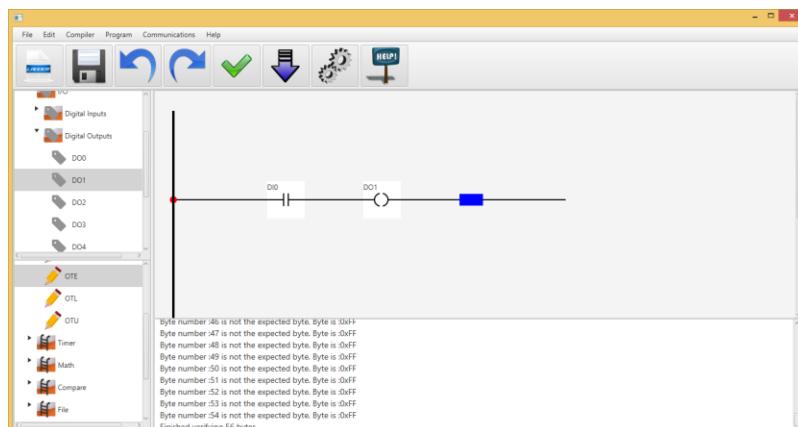


Figure 5.2.4 Ladder program downloaded to PLC with unsuccessful verification

The received data shows that the PLC was acknowledging each command, that is the third byte represents the acknowledgement and is 0x06 in all cases. It also shows that the erase (0x05), connect (0x06) and the flash write (0x01) commands were all acknowledged. The program read back from the PLC is all 0xFF's which are blank memory areas.

Affordable PLC Emulator For Students by Vusumuzi Tshabangu N0110328Z

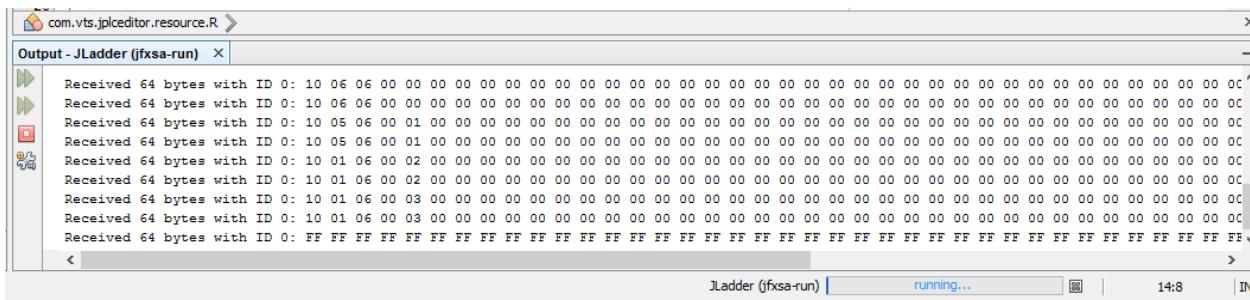


Figure 5.2.5 USB data received by desktop application displayed on Netbeans IDE output

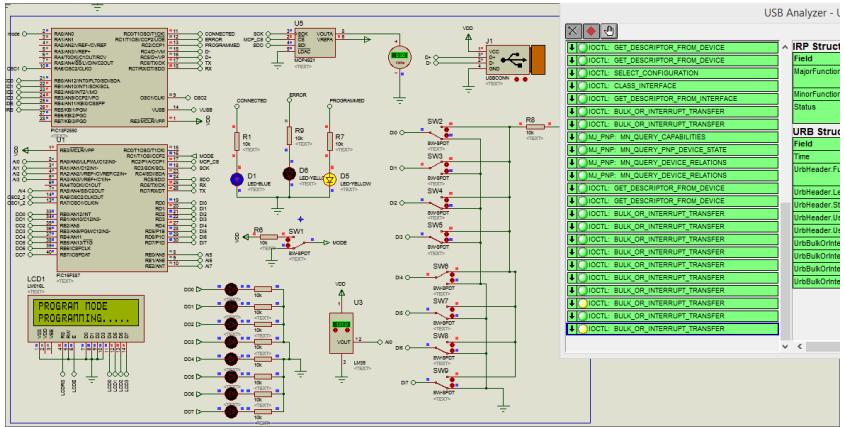


Figure 5.2.6 Proteus simulation for program mode

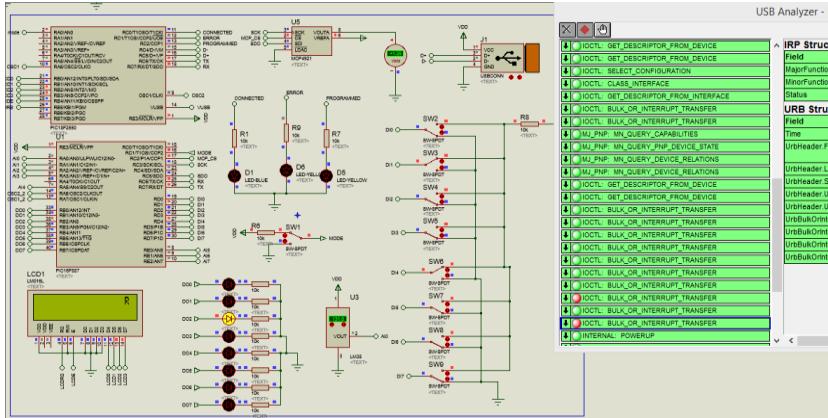


Figure 5.2.7 Proteus simulation after switching to run mode

The USB data being transmitted between through the USB data analyzer also showed that the proteus simulation is out of sync with what was happening, as the read program appears before the command to read the flash memory is sent.

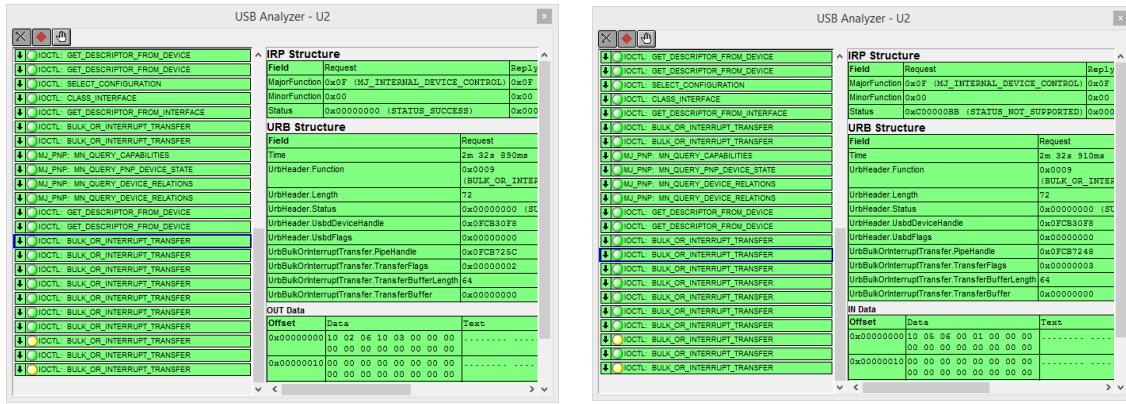


Figure 5.2.8 Connection request packet and the response

The first data sent by the application was a request to connect which is

<DLE(0x10)><STX(0x02)><CMD(0x06)><DLE(0x10)><ETX(0x03)>

The response was

<DLE(0x10)><CMD(0x06)><STS(0x06)><TID(0x0001)>

Which shows that the connection request was acknowledged by the emulator. The next desktop application request was to perform an erase of 128 bytes starting from address 0x2000 indicated by

<DLE(0x10)><STX(0x02)><CMD(0x05)><ADDR(0x2000)><LEN(0x02)><DLE(0x02)><ETX(0x03)>

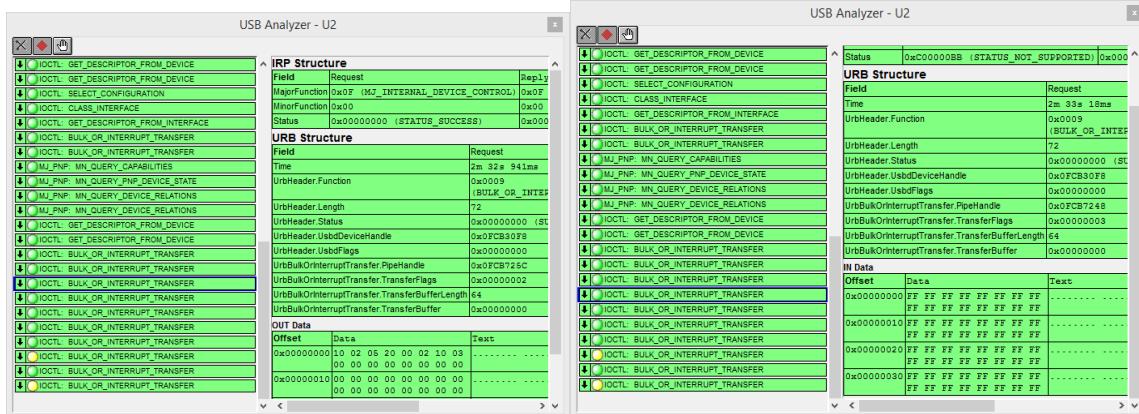


Figure 5.2.9 Flash Erase request packet and PLC response

In this case, the expected response was <DLE(0x10)><CMD(0x05)><STS(0x06)><TID(0x0002)>

Which is what is shown the application received from the output printed out. But the figure above shows the response was a chain of 0xFF's.

Affordable PLC Emulator For Students by Vusumuzi Tshabangu N0110328Z

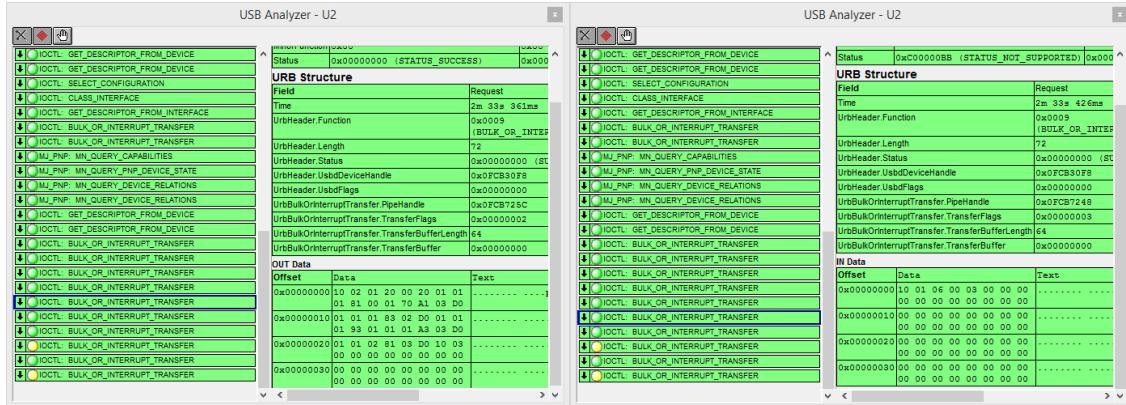


Figure 5.3.0 Flash write packet successfully acknowledged

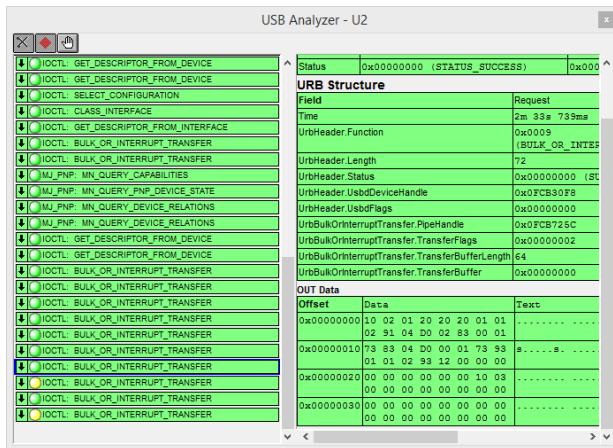


Figure 5.3.1 Second flash write packet not acknowledged

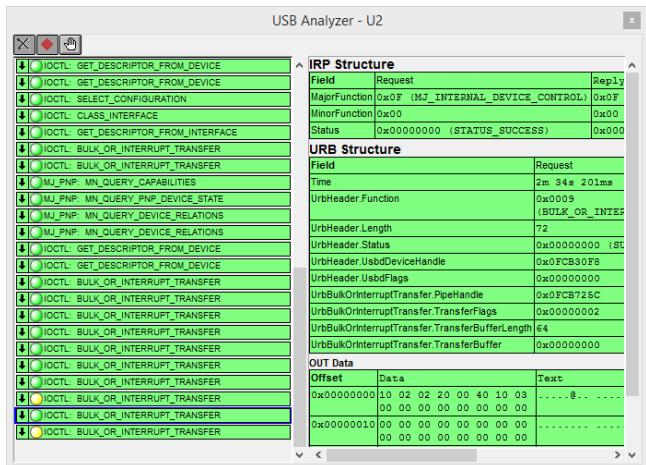


Figure 5.3.2 Flash read packet not acknowledged

Conclusion

The PLC emulator and desktop application were both communicating successfully but however, the results show inconclusive evidence as to whether the program was written to flash memory.

However, the same practical was repeated on hardware and there were no reading errors from the desktop application. Upon switching the PLC to run mode, the program ran successfully.

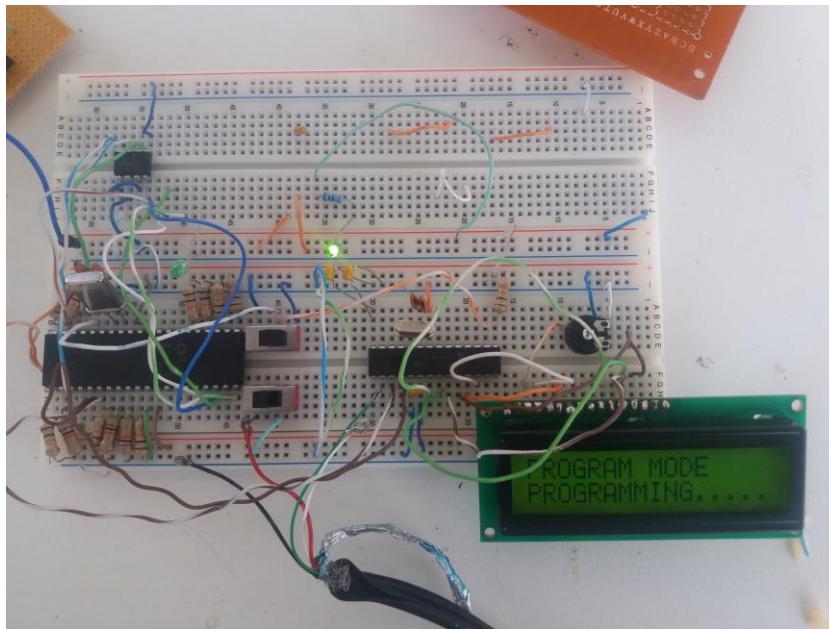


Figure 5.3.3 Processor board loading program on a bread board

5.3 Opto-isolator boards Implementation and Testing

This section looks at the simulation performed on the digital input and digital output boards.

5.3.1 Digital Input Circuit Simulation

Aim

To determine if the digital input opto-isolator circuit of figure 4.4 will isolate the CPU from 24V DC.

Method

1. Apply 24V DC to the input of the circuit.
2. Measure the output voltage with a voltmeter.

Results and analysis

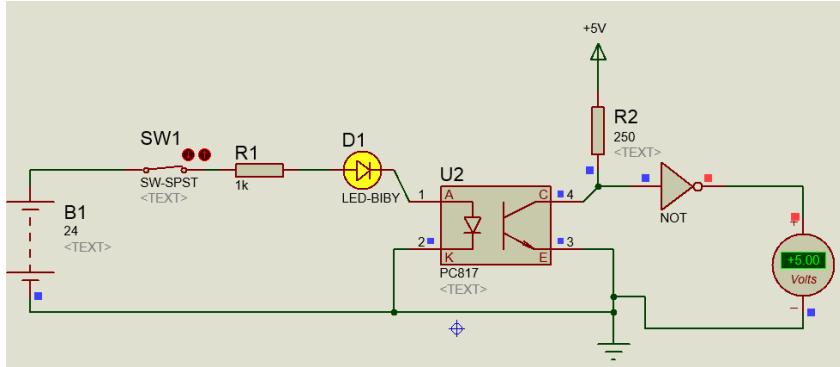


Figure 5.3.4 Digital Input circuit with 24V applied

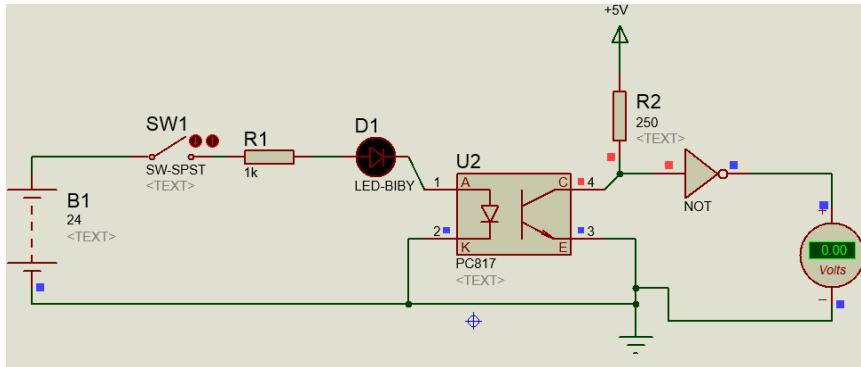


Figure 5.3.5 Digital Input opto-isolator circuit with 0V applied

Conclusion

The digital input opto-isolator board is able to isolate the CPU board from 24V DC.

5.3.2 Digital Output Circuit Simulation

Aim

To determine if the digital output opto-isolator circuit of figure 4.5 will generate 24V for 5V input.

Method

1. Apply 5V DC to the input of the circuit.
2. Measure the output voltage with a voltmeter.

Results and analysis

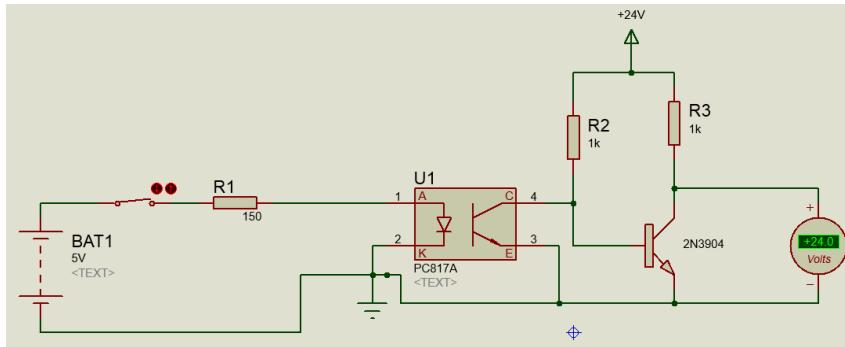


Figure 5.3.4 Digital Output circuit with 5V applied

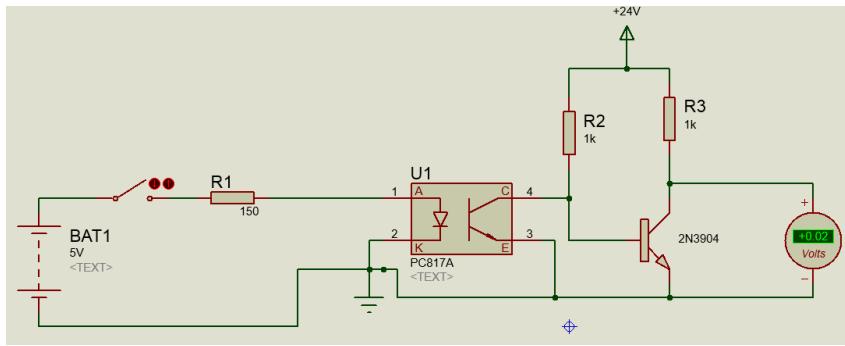


Figure 5.3.5 Digital output opto-isolator circuit with 0V applied

Conclusion

The digital output opto-isolator board is able to generate the 24V DC.

Chapter 6: Conclusion

This Chapter looks at the author's evaluation of the project.

6.1 Objectives Met

The final prototype for the PLC emulator was successfully transferred onto a prototype board and interfaced to the training kit as shown in figure 6.1 below.

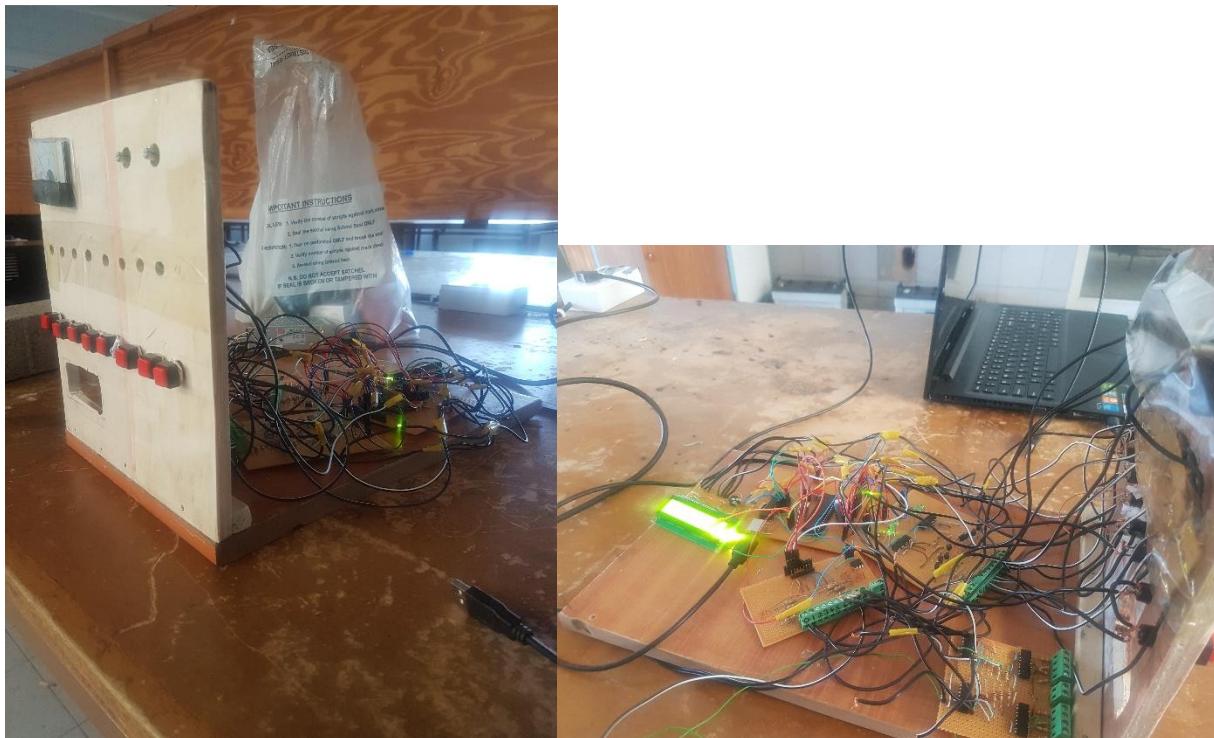


Figure 6.1 Final PLC emulator prototype

1. The desktop application that allows the user to develop ladder program, compile it and download it to the PLC emulator was successfully developed. The desktop application has its own data structure which simplifies variable or tag creation without the user being technically able.
2. The Microcontroller operating system was successfully developed that allows the PLC Emulator to communicate with the desktop application, load the PLC program and run it on a PLC training kit.
3. The entire PLC emulator was successfully assembled consisting of:
 - a. A power supply that provides both 5V and 24V DC.
 - b. Digital input and digital output opto-isolator boards.
 - c. The CPU board.
 - d. The PLC training, I/O interface.

Hence the primary goal of the project was met.

6.2 Problem Statement

The problem this PLC emulator was attempting to solve was cost and table 6 clearly shows that this emulator is very affordable to build. PLCs are generally expensive and the goal of this project was to design an affordable PLC emulator and its desktop compiler that is better than the free software LDmicro. The developed compiler complies with the IEC 61131-3 specification for a PLC program and offers a better GUI compared to the LDmicro software but the only disadvantage it has is its inability to produce branches within the Ladder diagram.

Component	Price/USD
Pic18F2550	7
Pic16F887	4
CNY74-4 x4	5
MCP4921	5
VeroBoards	20
Terminal Blocks	6
Power Supply	25
Soldering wire	5
Other Components	12
TOTAL	79

Table 6 Project budget for a single PLC unit

The author was aware of this need but due to the excessive work of the project and the limited amount of time it had, the author was not able to fully test the method of both drawing the branch on the ladder diagram and compiling it.

6.3 Recommendations

The PLC emulator only has one analogue output, but it can have more than one by having an SPI DAC chip with more than one output like the MCP4922 which has two analogue outputs. This was the DAC that was expected to be purchased but was out of stock from Mantech South Africa.

The author had also bought an 8 channel MAX521ACPP+ DAC with a resolution of 8 bits. But because of the lack of time to test and integrate the DAC both at hardware and software level, it was left out.

Instead of having a PIC microcontroller as the I/O interface microcontroller, the author could have used port expanders like the MCP23S17 16 bit SPI port expander with two general purpose I/O ports namely PORTA and PORTB and three analog inputs. More such port expanders are available which could be connected as slaves to the user program microcontroller and allow for expansion card features.

References

1. **Programmable Logic Controller, Wikipedia**
https://en.wikipedia.org/wiki/Programmable_logic_controller
2. **MICROLOGIX PROGRAMMABLE CONTROLLERS SELECTION GUIDE**
3. **OpenPLC Overview** <http://www.openplcproject.com/>
4. **LD Micro overview and features:** <http://cq.cx/ladder.pl>
5. JavaFX Getting Started with JavaFX, Release 8, E50607-02, Oracle
6. **JavaFX Working with the JavaFX Scene Graph, Release 8, E50683-01**
7. **JavaFX Mastering FXML, Release 8, E50650-01**
8. **JavaFX Handling Events, Release 8, E50628-01**
9. mikroc_pic_pro_manual_v101
10. In-Circuit Serial Programming™ (ICSP™) Guide
11. PIC18F2455/2550/4455/4550 Data Sheet
12. **MCP4921/4922 12-Bit DAC with SPI™ Interface data sheet**
13. PIC16F882/883/884/886/887 Data Sheet
14. Universal Serial Bus (USB), Device Class Definition for Human Interface Devices (HID) Firmware Specification—6/27/01, Version 1.11
15. UML Package Diagrams Notation : <http://www.uml-diagrams.org/package-diagrams.html>
16. JavaFX Scene Builder:
<http://www.oracle.com/technetwork/java/javase/downloads/javafxscenefuilder-info-2157684.html>
17. Xstream Java Library : <http://x-stream.github.io/>
18. DF1 Protocol and Command Set Reference Manual
http://literature.rockwellautomation.com/idc/groups/literature/documents/rm/1770-rm516_en-p.pdf
19. UHB Protocol specification <https://github.com/sergev/pic32prog/wiki/UHB-Protocol>
20. INTERNATIONAL STANDARD IEC 61131-1 Second edition 2003-05
21. Algorithm for Compiling Unrestricted Ladder Diagram to IEC 61131-3 Compliant Instruction List.
Proceedings of the World Congress on Engineering 2011 Vol II WCE 2011, July 6 - 8, 2011,
London, U.K. by Kando Hamiyanze Moonga, Member IAENG, You Linru, and Liu Shaojun
22. Rockwell Automation Literature library:
https://www.rockwellautomation.com/en_IN/literature-library/overview.page?

Appendix A: User program microcontroller code

```
#include <built_in.h>
//== DF! Protocol Constants
=====
//Protocol is currently not fully implemented
const DLE = 0x10;
const STX = 0x02;
const ACK = 0x06;
const NAK = 0x0F;
const ETX = 0x03;
//== Encapsulation Protocol commands
=====
enum Command {
    cmdFREE, cmdWRITE_FLASH, cmdREAD_FLASH, cmdWRITE_RAM, cmdREAD_RAM,
    cmdERASE, cmdCONNECT
};
//== all the functions
=====
void runMode();
void programMode();
void decodeData();
void sendAck();
void userProgram() org 0x2000;
void checkUsb();
void serviceRunCmd();
//Variables will start at 0x60
//== Local I/O image variables
=====
unsigned int anIn0 absolute 0x60;
unsigned int anIn1 absolute 0x62;
unsigned int anIn2 absolute 0x64;
unsigned int anIn3 absolute 0x66;
unsigned int anIn4 absolute 0x68;
unsigned int anIn5 absolute 0x6A;
unsigned int anIn6 absolute 0x6C;
unsigned int anIn7 absolute 0x6E;
char dii absolute 0x70;
unsigned int anOut absolute 0x71;
char dio absolute 0x73;
//array with the order of all Inputs and Outputs
unsigned short io[20] absolute 0x60;
//== USB variables and buffer
=====
unsigned char readbuff[64] absolute 0x500; // USB Buffers in Bank 6
unsigned char writebuff[64] absolute 0x540;
unsigned char Reserve5thBankForUSB[256] absolute 0x400; // Dummy allocation
of 5th bank
// (used by USB module internally), to prevent compiler from allocating it
unsigned short length absolute 0x74; //length of data section of protocol
unsigned int address absolute 0x75;
char flags absolute 0x77;
unsigned short state absolute 0x78;
unsigned int transactionID absolute 0x79;
```

```

sbit programming at flags.b0;
//===== Other variables
=====
short session absolute 0x7B;
short tempByte absolute 0x7C;
short i absolute 0x7D, k absolute 0x7E;
short ramAddress absolute 0x7F;
short ramByte absolute 0x80;
char rxByte;
enum Command cmd;
//===== LCD Initialization
=====
// Lcd pinout settings
sbit LCD_RS at RB5_bit;
sbit LCD_EN at RB4_bit;
sbit LCD_D7 at RB3_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D4 at RB0_bit;

// Pin direction
sbit LCD_RS_Direction at TRISB5_bit;
sbit LCD_EN_Direction at TRISB4_bit;
sbit LCD_D7_Direction at TRISB3_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D4_Direction at TRISB0_bit;
//==== I/O connections
=====
sbit connected at rc0_bit;
sbit programmed at rc2_bit;
sbit plc_mode at ra0_bit;

sbit connected_dir at trisc0_bit;
sbit programmed_dir at trisc2_bit;
sbit plc_mode_dir at trisa0_bit;
//==== Other constants
=====
const unsigned int OS_SIZE = 8022; //byte size of the operationg system
const unsigned int PROGRAM_START = 8192; //0x2000
//== Reserved Banks for user program to prevent compiler from using them
=====
unsigned char Reserved1stBankForOS[256] absolute 0x00;
unsigned char Reserved2ndBankForUser[256] absolute 0x100;
unsigned char Reserved3rdBankForUser[256] absolute 0x200;

void interrupt() {
    //USB interrupt procedure.
    USB_Interrupt_Proc();
    return;
}

void main() {
    uart1_init(9600);
    delay_ms(100);
    //---- I/O configuration -----
-----

```

```

ADCON1 |= 0x0F; // Configure all ports with analog function as digital
CMCON |= 7; // Disable comparators

plc_mode_dir = 1; //plc mode pin is input
cmd = cmdFREE; //free cmd

portb = 0;
trisb = 0; //lcd port
connected_dir = 0; //connected pin is output
programmed_dir = 0;
connected = 0;
for (k = 0; k < 2; k++) {
    if (flash_read(0x2007 + k) == 0xFF)
        programmed = 0;
    else
        programmed = 1;
}

----- Modules Initialization -----
-----
Lcd_Init(); // Initialize LCD
Lcd_Cmd(_LCD_CLEAR); // Clear display
Lcd_Cmd(_LCD_CURSOR_OFF); // Cursor off
//USB Initialization
HID_Enable(&readbuff, &writebuff); // Enable HID communication
delay_ms(100); //wait for initialization to complete
//Program loop
while (1) {
    lcd_cmd(_lcd_clear);
    if (plc_mode) {
        lcd_out(1, 1, "RUN MODE");
        runMode();
    } else {
        lcd_out(1, 1, "PROGRAM MODE");
        programMode();
    }
}
}

void runMode() {
run_loop:
    //Read Inputs
    for (session = 0; session < 17; session++) { //read 17 bytes worth of
input data
        uart1_write(cmdREAD_RAM); //send request for data to be read
        delay_ms(5);
        if (uart1_data_ready())
            io[session] = uart1_read();
    }
    //Run program
    if (programmed)
        userProgram();
    //Write outputs
    for (session = 17; session < 20; session++) {
        if (UART1_Tx_Idle()) //if transmission to finished
            uart1_write(io[session]);
        delay_ms(5);
    }
}

```

```

    }
    //LCD function
    lcd_cmd(_lcd_shift_right);
    checkUsb();
    if (rxByte)
        serviceRunCmd();
    delay_ms(100);
    if (plc_mode) goto run_loop;
    //else switch to program mode.
    return;
}

void programMode() {
program_loop:// program mode loop starts here
    checkUsb();
    if (rxByte) { // Do we have an incoming?
        switch (cmd) { // Process command.
            case cmdERASE: // Cmd: Disconnect PLC and PC app.
            {
                state = ACK;
                for (k = 0; k < length; k++) {
                    if (address >= PROGRAM_START) {
                        Flash_Erase_64(address);
                        address += 64;
                    } else {
                        state = NAK;
                    }
                }
                ++transactionID;
                sendAck();
                cmd = cmdFREE;
                break;
            }
            case cmdCONNECT: // Cmd: Connect PLC and PC app.
            {
                connected = 1;
                lcd_out(2, 1, "CONNECTED TO PC.");
                state = ACK;
                transactionID = 0;
                sendAck();
                cmd = cmdFREE; // Set 'Idle' command code.
                break;
            }
            case cmdWRITE_FLASH:
            { // Cmd: Write to program memory.
                state = NAK;
                if (address >= PROGRAM_START) { //If received address is
inside OS
                    FLASH_Write_32(address, &readbuff[6]); //write data to
Flash
                    lcd_out(2, 1, "PROGRAMMING.....");
                    state = ACK;
                    programmed = 1;
                }
                ++transactionID;
                sendACK();
                cmd = cmdFREE;
            }
        }
    }
}

```

```

        break;
    }
    case cmdREAD_FLASH:
    { // Cmd: Read from program memory.
        if (address >= PROGRAM_START) { //only read user program, not
OS
            FLASH_Read_N_Bytes(address, &writebuff[0], 64); // read
64 bytes starting from address
            HID_Write(&writebuff[0], 64); // Send data.
        } else {
            state = NAK;
            ++transactionID;
            sendAck();
        }
        cmd = cmdFREE; // Set 'Idle' command code
        break;
    }
}
delay_ms(100);
if (!plc_mode) goto program_loop; //else disable HID device and go back
to run mode
return;
}

void serviceRunCmd() {
    switch (cmd) { // Process command.
        case cmdREAD_RAM:
        { // Cmd: Read from program memory.
            asm movff _address, fsr0;
            for (i = 0; i < length; i++) {
                asm movff indf0, _ramByte;
                writebuff[i] = ramByte;
                asm incf fsr0;
            }
            HID_Write(&writebuff[0], 64); // Send data.
            cmd = cmdFREE; // Set 'Idle' command code
            break;
        }
        case cmdWRITE_RAM:
        { // Cmd: Read from program memory.
            asm movff _address, fsr0;
            for (i = 0; i < length; i++) {
                ramByte = readbuff[i+6];
                asm movff _ramByte, indf0;
                asm incf fsr0;
            }
            state = ACK;
            ++transactionID;
            sendAck();
            cmd = cmdFREE; // Set 'Idle' command code
            break;
        }
    }
return;
}

```

```
void checkUsb() {
    USB_Polling_Proc(); // Check USB for new data.
    rxByte = HID_Read();
    if (rxByte && cmd == cmdFREE) { // Do we have an incoming?
        // Check received packet for new command.
        // Only decode if we in 'Idle' mode
        if (readbuff[0] != DLE) { // Do we have a 'DLE' at start?
            return;
        }
        if (readbuff[1] != STX) { // Do we have a 'STX'?
            return; // received data is not protocol, return.
        }

        // Process received command if first two bytes are protocol.
        cmd = readbuff[2]; // Get command code.
        Hi(address) = readbuff[3]; // Get address hi byte.
        Lo(address) = readbuff[4]; // Get address low byte.
        length = readBuff[5]; // Get counter byte
    }
    return;
}

void sendAck() {
    // Make acknowledgment packet.
    writebuff[0] = DLE; // Start of packet identifier.
    writebuff[1] = cmd; // Command code to acknowledge.
    writebuff[2] = state; //acknowledgment status
    writebuff[3] = Hi(transactionID); //message number
    writebuff[4] = Lo(transactionID);
    for (i = 5; i < 64; i++) {
        writebuff[i] = 0;
    }
    HID_Write(&writebuff[0], 64); // Send acknowledgment packet..
    return;
}

void userProgram() org 0x2000 {
```

Appendix B: I/O interface microcontroller code

```
#include <built_in.h>
//I/O variables
unsigned int anIn0 ;
unsigned int anIn1 ;
unsigned int anIn2 ;
unsigned int anIn3 ;
unsigned int anIn4 ;
unsigned int anIn5 ;
unsigned int anIn6 ;
unsigned int anIn7 ;
unsigned int anOut;
//uart variables
unsigned short rxByte,txByte;
unsigned short temp,session;
//DAC SPI pins
sbit mcp_cs at rc2_bit;
sbit mode at rc1_bit;

sbit mcp_cs_dir at trisc2_bit;
sbit mode_dir at triscl_bit;

//ISR
void interrupt() {
    if(pir1.rcif){ //Uart receive interrupt
        rxByte = rcreg; //Clear interrupt by reading receive register
        ++session;
        //The first USART session is reading of inputs

        if(session==1){ //next session if reading of analog inputs
            txByte= Lo(anIn0);
            uart1_write(txByte);
            return;
        }
        else if(session==2){
            txByte=Hi(anIn0);
            uart1_write(txByte);
            return;
        }
        else if(session==3){ //next session if reading of analog inputs
            txByte= Lo(anIn1);
            uart1_write(txByte);
            return;
        }
        else if(session==4){
            txByte=Hi(anIn1);
            uart1_write(txByte);
            return;
        }
        else if(session==5){ //next session if reading of analog inputs
            txByte= Lo(anIn2);
            uart1_write(txByte);
            return;
        }
        else if(session==6){
```

```
txByte=Hi(anIn2);
uart1_write(txByte);
return;
}
else if(session==7){//next session if reading of analog inputs
txByte= Lo(anIn3);
uart1_write(txByte);
return;
}
else if(session==8){
txByte=Hi(anIn3);
uart1_write(txByte);
return;
}
else if(session==9){//next session if reading of analog inputs
txByte= Lo(anIn4);
uart1_write(txByte);
return;
}
else if(session==10){
txByte=Hi(anIn4);
uart1_write(txByte);
return;
}
else if(session==11){//next session if reading of analog inputs
txByte= Lo(anIn5);
uart1_write(txByte);
return;
}
else if(session==12){
txByte=Hi(anIn5);
uart1_write(txByte);
return;
}
else if(session==13){//next session if reading of analog inputs
txByte= Lo(anIn6);
uart1_write(txByte);
return;
}
else if(session==14){
txByte=Hi(anIn6);
uart1_write(txByte);
return;
}
else if(session==15){//next session if reading of analog inputs
txByte= Lo(anIn7);
uart1_write(txByte);
return;
}
else if(session==16){
txByte=Hi(anIn7);
uart1_write(txByte);
return;
}
else if(session==17){ //Digital inputs are read first
txByte= portd;
uart1_write(txByte);
```

```

        return;
    }
    //reading sessions ends here. Then comes writing session

    else if(session==18){
        Lo(anOut)=rxByte;
        return;
    }
    else if(session==19){
        Hi(anOut)=rxByte;

        return;
    }
    else if(session==20){ //first data to be written is digital output
        portb=rxByte;
        session=0;
        return;
    }
    return;
}
}

void main() {
    //I/O Configuration
    CCP1CON = 0;      // Disable comparators
    //Configure channels 0 to 7 as analog, the rest as digital
    ANSEL = 0xFF;    // Configure AN0- AN7 pins as analog I/O
    ANSELH = 0;       //configure AN8-AN13 as digital
    C1ON_bit = 0;     // Disable comparators
    C2ON_bit = 0;
    option_reg.F7=1; //enable pull up resistors bit
    WPUB=0xFF; //enable pull up resistors pins
    TRISA = 0x2F;   // analog pins as inputs
    mode_dir =1;    //plc mode pin is input
    PORTD =0;
    TRISD = 0xFF;   //Digital Inputs port as inputs
    portb = 0;
    trisb =0x00;    //Digital outputs port as outputs
    mcp_cs_dir=0;   //MCP Chip Select pin is output
    mcp_cs=1;        //Disable MCP DAC

    //Uart configuration
    intcon.gie=1;    //globally enable all interrupts
    intcon.peie=1;   //Enables all unmasked peripheral interrupts
    pie1.rcie=1;     //Enables the EUSART receive interrupt
    UART1_Init(9600); //init uart at 9600 bits per second
    delay_ms(100);   //wait for uart initialization

    adc_init();       //initialize ADC
    delay_ms(10);
    //SPI Initialization
    SPI1_Init();
    delay_ms(100);
    //reset some counters
    session=0;
    anOut=0;
}

```

```
//the loop code:  
do {  
    if(mode){ //Only if we are in run mode  
        anIn0 = ADC_Read(0); // Get 10-bit result of AD conversion  
        anIn1 = ADC_Read(1); // Get 10-bit result of AD conversion  
        anIn2 = ADC_Read(2); // Get 10-bit result of AD conversion  
        anIn3 = ADC_Read(3); // Get 10-bit result of AD conversion  
        anIn4 = ADC_Read(4); // Get 10-bit result of AD conversion  
        anIn5 = ADC_Read(5); // Get 10-bit result of AD conversion  
        anIn6 = ADC_Read(6); // Get 10-bit result of AD conversion  
        anIn7 = ADC_Read(7); // Get 10-bit result of AD conversion  
  
        mcp_cs = 0; //Select MCP4921 DAC  
        temp = Hi(anOut)&0xF; // Store valueDAC[11..8] to temp[3..0]  
        temp |= 0x30; // Define DAC setting, see MCP4921 datasheet  
        SPI1_Write(temp); // Send high byte via SPI  
        // Send Low Byte  
        temp = Lo(anOut);  
        SPI1_Write(temp); // Send low byte via SPI  
        mcp_cs = 1; //Disable DAC  
        delay_ms(100);  
    }  
} while(1);  
}
```

Appendix C: Run Mode Test user program microcontroller code

```
void runMode();
void readInputs();
void writeOutputs();
void userProgram();
void testMode();
void myISR();

//===== Local I/O image variables
=====
unsigned int anIn0 absolute 0x60;
unsigned int anIn1 absolute 0x62;
unsigned int anIn2 absolute 0x64;
unsigned int anIn3 absolute 0x66;
unsigned int anIn4 absolute 0x68;
unsigned int anIn5 absolute 0x6A;
unsigned int anIn6 absolute 0x6C;
unsigned int anIn7 absolute 0x6E;
char dii absolute 0x70;
unsigned int anOut absolute 0x71;
char dio absolute 0x73;
//array with the order of all Inputs and Outputs
unsigned short io[20] absolute 0x60;

char line1[8];
char line2[8];
short rxByte, i, txByte;
bit dataReady;

sbit ss at rb2_bit;
sbit plc_mode at ra0_bit;

sbit ss_dir at trisb2_bit;
sbit plc_mode_dir at trisa0_bit;

// Lcd pinout settings
sbit LCD_RS at RB5_bit;
sbit LCD_EN at RB4_bit;
sbit LCD_D7 at RB3_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D4 at RB0_bit;

// Pin direction
sbit LCD_RS_Direction at TRISB5_bit;
sbit LCD_EN_Direction at TRISB.b4;
sbit LCD_D7_Direction at TRISB.B3;
sbit LCD_D6_Direction at TRISB.B2;
sbit LCD_D5_Direction at TRISB.B1;
sbit LCD_D4_Direction at TRISB.B0;

void main() {
```

```

ADCON1 |= 0x0F; // Configure all ports with analog function as digital
CMCON |= 7; // Disable comparators

plc_mode_dir = 1;
anIn0=0;
anOut=0;
portb=0;
trisb=0;
Lcd_Init(); // Initialize LCD
Lcd_Cmd(_LCD_CLEAR); // Clear display
Lcd_Cmd(_LCD_UNDERLINE_ON); // Cursor off
Lcd_Out(1, 1, "ADC Value:"); // Write text in first row
Lcd_Out(2, 1, "DAC Value:"); // Write text in first row
delay_ms(500);
Uart1_Init(9600);
delay_ms(100);
while (1) {
    runMode();
}
}

void runMode() {
while(plc_mode){
    //Read Inputs
    for (i = 0; i < 17; i++) { //read 17 bytes worth of input data
        uart1_write(2); //send request for data to be read
        delay_ms(10);
        if (uart1_data_ready())
            io[i] = uart1_read();
    }
    //Run program
    userProgram();
    //Write outputs
    for (i = 17; i < 20; i++) {
        if (UART1_Tx_Idle()) //wait for transmission to finish
            uart1_write(io[i]);
        delay_ms(10);
    }
    IntToStr(anIn0,line1);
    IntToStr(anOut,line2);
    lcd_out(1,11,line1);
    lcd_out(2,11,line2);
    delay_ms(200);
    } //else switch to program mode.
    return;
}

void userProgram() {
    dio = dii;
    anOut = anIn0;
    return;
}

```

Appendix D: JPLC Editor main package classes code

```
package com.vts.jplceditor;

import com.vts.jplceditor.resource.R;
import java.io.IOException;
import java.net.URL;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.geometry.Rectangle2D;
import javafx.scene.Scene;
import javafx.scene.layout.VBox;
import javafx.stage.Screen;
import javafx.stage.Stage;

public class JPlcEditor extends Application {

    URL url = null;
    VBox root = null;
    Rectangle2D screenBounds;

    @Override
    public void start(Stage primaryStage) {
        url = getClass().getResource(R.WINDOW_MAIN);
        screenBounds = Screen.getPrimary().getVisualBounds();
        try {
            FXMLLoader loader = new FXMLLoader(url);
            root = loader.load();

            Scene scene = new Scene(root);
            primaryStage.setX(screenBounds.getMinX());
            primaryStage.setY(screenBounds.getMinY());
            primaryStage.setWidth(screenBounds.getWidth());
            primaryStage.setHeight(screenBounds.getHeight());
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch (IOException ex) {
            Logger.getLogger(JPlcEditor.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }
}
```

Appendix : JPLC Editor user interaction layer classes code

MainWindow.fxml code

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.*?>
<?import javafx.scene.text.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<VBox maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity" prefHeight="600.0" prefWidth="800.0"
stylesheets="@css/mainwindow.css" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.vts.jplceditor.ui.controller.MainWindowController">
    <children>
        <MenuBar>
            <menus>
                <Menu mnemonicParsing="false" text="File">
                    <items>
                        <MenuItem mnemonicParsing="false"
onAction="#createNewProject" text="New Project" />
                        <MenuItem mnemonicParsing="false" onAction="#openProject"
text="Open Project" />
                        <RadioMenuItem mnemonicParsing="false"
onAction="#saveProject" text="Save Project" />
                        <MenuItem mnemonicParsing="false" onAction="#onExit"
text="Exit" />
                    </items>
                </Menu>
                <Menu mnemonicParsing="false" text="Edit">
                    <items>
                        <MenuItem mnemonicParsing="false" onAction="#undoAction"
text="Undo" />
                        <MenuItem mnemonicParsing="false" onAction="#redoAction"
text="Redo" />
                        <MenuItem mnemonicParsing="false" onAction="#deleteAction"
text="Delete" />
                    </items>
                </Menu>
                <Menu mnemonicParsing="false" text="Compiler">
                    <items>
                        <MenuItem mnemonicParsing="false" onAction="#verifyProgram"
text="Verify" />
                        <MenuItem mnemonicParsing="false"
onAction="#compileProgram" text="Compile" />
                    </items>
                </Menu>
                <Menu mnemonicParsing="false" text="Program">
                    <items>
                        <MenuItem mnemonicParsing="false" onAction="#saveProgram"
text="Save Program" />
                        <Menu mnemonicParsing="false" text="Create">
                            <items>
```

```
<MenuItem mnemonicParsing="false" onAction="#createTag"
text="Tag" />
<MenuItem mnemonicParsing="false"
onAction="#createRung" text="Rung" />
<MenuItem mnemonicParsing="false"
onAction="#createNewLadder" text="Program" />
</items>
</Menu>
</items>
</MenuBar>
<ToolBar prefHeight="80.0">
<items>
<Button fx:id="newLadderButton" mnemonicParsing="false"
onAction="#createNewProject">
<graphic>
<ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
<image>
<Image url="@../icon/new_ladder-icon.png" />
</image>
</ImageView>
</graphic>
<tooltip>
<Tooltip text="New Ladder Program" />
</tooltip>
</Button>
<Button fx:id="saveButton" mnemonicParsing="false"
onAction="#saveProject">
<graphic>
<ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
<image>
<Image url="@../icon/save-icon.png" />
</image>
</ImageView>
</graphic>
<tooltip>
<Tooltip text="Save Ladder Program" />
</tooltip>
</Button>
```

```
<Button fx:id="undoButton" mnemonicParsing="false"
onAction="#undoAction">
    <graphic>
        <ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/undo-icon.png" />
            </image>
        </ImageView>
    </graphic>
    <tooltip>
        <Tooltip text="Undo" />
    </tooltip>
</Button>
<Button fx:id="redoButton" mnemonicParsing="false"
onAction="#redoAction">
    <graphic>
        <ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/redo-icon.png" />
            </image>
        </ImageView>
    </graphic>
    <tooltip>
        <Tooltip text="Redo" />
    </tooltip>
</Button>
<Button fx:id="verifyButton" mnemonicParsing="false"
onAction="#verifyProgram">
    <graphic>
        <ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/ok-apply-icon.png" />
            </image>
        </ImageView>
    </graphic>
    <tooltip>
        <Tooltip text="Verify Program" />
    </tooltip>
</Button>
<Button fx:id="downloadButton" mnemonicParsing="false"
onAction="#downloadProgram">
    <graphic>
        <ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/download-icon.png" />
            </image>
        </ImageView>
    </graphic>
    <tooltip>
        <Tooltip text="Download Program" />
    </tooltip>
</Button>
```

```
<Button fx:id="settingsButton" mnemonicParsing="false"
onAction="#settingsAction">
    <graphic>
        <ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/settings-icon.png" />
            </image>
        </ImageView>
    </graphic>
    <tooltip>
        <Tooltip text="Settings" />
    </tooltip>
</Button>
<Button fx:id="redoButton1" mnemonicParsing="false"
onAction="#helpAction">
    <graphic>
        <ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/help-icon.png" />
            </image>
        </ImageView>
    </graphic>
    <tooltip>
        <Tooltip text="Help" />
    </tooltip>
</Button>
</items>
</ToolBar>
<SplitPane dividerPositions="0.33" minHeight="490.0" minWidth="800.0"
nodeOrientation="LEFT_TO_RIGHT" VBox.vgrow="ALWAYS">
    <items>
        <VBox minHeight="490.0" minWidth="200.0">
            <children>
                <ScrollPane minHeight="200.0" minWidth="200.0">
                    <content>
                        <TreeView fx:id="programTree" minHeight="600.0"
minWidth="200.0" onDragDetected="#onProgramDragDetected" />
                    </content>
                </ScrollPane>
                <ScrollPane minHeight="289.0" minWidth="200.0">
                    <content>
                        <TreeView fx:id="instructionTree" minHeight="600.0"
minWidth="200.0" onDragDetected="#onInstructionDragDetected" />
                    </content>
                </ScrollPane>
            </children>
        </VBox>
        <VBox minHeight="490.0" minWidth="200.0">
            <children>
                <ScrollPane fx:id="ladderPane" minHeight="390.0"
minWidth="200.0" />
                    <TextArea fx:id="status" editable="false" minHeight="200.0"
minWidth="200.0" wrapText="true">
                        <font>
                            <Font size="14.0" />

```

```
        </font></TextArea>
    </children>
</VBox>
</items>
<VBox.margin>
    <Insets />
</VBox.margin>
</SplitPane>
</children>
</VBox>

```

TagWindow.fxml code

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.*?>
<?import javafx.scene.text.*?>
<?import java<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<VBox maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity" prefHeight="600.0" prefWidth="800.0"
stylesheets="@css/mainwindow.css" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.vts.jplceditor.ui.controller.MainWindowController">
    <children>
        <MenuBar>
            <menus>
                <Menu mnemonicParsing="false" text="File">
                    <items>
                        <MenuItem mnemonicParsing="false"
onAction="#createNewProject" text="New Project" />
                        <MenuItem mnemonicParsing="false" onAction="#openProject"
text="Open Project" />
                        <RadioMenuItem mnemonicParsing="false"
onAction="#saveProject" text="Save Project" />
                        <MenuItem mnemonicParsing="false" onAction="#onExit"
text="Exit" />
                    </items>
                </Menu>
                <Menu mnemonicParsing="false" text="Edit">
                    <items>
                        <MenuItem mnemonicParsing="false" onAction="#undoAction"
text="Undo" />
                        <MenuItem mnemonicParsing="false" onAction="#redoAction"
text="Redo" />
                        <MenuItem mnemonicParsing="false" onAction="#deleteAction"
text="Delete" />
                    </items>
                </Menu>
                <Menu mnemonicParsing="false" text="Compiler">
                    <items>
```

```

        <MenuItem mnemonicParsing="false" onAction="#verifyProgram"
text="Verify" />
        <MenuItem mnemonicParsing="false"
onAction="#compileProgram" text="Compile" />
    </items>
</Menu>
<Menu mnemonicParsing="false" text="Program">
    <items>
        <MenuItem mnemonicParsing="false" onAction="#saveProgram"
text="Save Program" />
        <Menu mnemonicParsing="false" text="Create">
            <items>
                <MenuItem mnemonicParsing="false" onAction="#createTag"
text="Tag" />
                <MenuItem mnemonicParsing="false"
onAction="#createRung" text="Rung" />
                <MenuItem mnemonicParsing="false"
onAction="#createNewLadder" text="Program" />
            </items>
        </Menu>
    </items>
</Menu>
<Menu mnemonicParsing="false" text="Communications">
    <items>
        <MenuItem mnemonicParsing="false" onAction="#downloadProgram"
text="Download" />
        <MenuItem mnemonicParsing="false"
onAction="#commSettingsAction" text="Settings" />
    </items>
</Menu>
<Menu mnemonicParsing="false" text="Help">
    <items>
        <MenuItem mnemonicParsing="false"
onAction="#documentationAction" text="Documentation" />
    </items>
</Menu>
</menus>
</MenuBar>
<ToolBar prefHeight="80.0">
    <items>
        <Button fx:id="newLadderButton" mnemonicParsing="false"
onAction="#createNewProject">
            <graphic>
                <ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
                    <image>
                        <Image url="@../icon/new_ladder-icon.png" />
                    </image>
                </ImageView>
            </graphic>
            <tooltip>
                <Tooltip text="New Ladder Program" />
            </tooltip>
        </Button>
        <Button fx:id="saveButton" mnemonicParsing="false"
onAction="#saveProject">
            <graphic>
```

```
<ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
    <image>
        <Image url="@../icon/save-icon.png" />
    </image>
</ImageView>
</graphic>
<tooltip>
    <Tooltip text="Save Ladder Program" />
</tooltip>
</Button>
<Button fx:id="undoButton" mnemonicParsing="false"
onAction="#undoAction">
    <graphic>
        <ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/undo-icon.png" />
            </image>
        </ImageView>
    </graphic>
    <tooltip>
        <Tooltip text="Undo" />
    </tooltip>
</Button>
<Button fx:id="redoButton" mnemonicParsing="false"
onAction="#redoAction">
    <graphic>
        <ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/redo-icon.png" />
            </image>
        </ImageView>
    </graphic>
    <tooltip>
        <Tooltip text="Redo" />
    </tooltip>
</Button>
<Button fx:id="verifyButton" mnemonicParsing="false"
onAction="#verifyProgram">
    <graphic>
        <ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/ok-apply-icon.png" />
            </image>
        </ImageView>
    </graphic>
    <tooltip>
        <Tooltip text="Verify Program" />
    </tooltip>
</Button>
<Button fx:id="downloadButton" mnemonicParsing="false"
onAction="#downloadProgram">
    <graphic>
```

```
<ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
    <image>
        <Image url="@../icon/download-icon.png" />
    </image>
</ImageView>
</graphic>
<tooltip>
    <Tooltip text="Download Program" />
</tooltip>
</Button>
<Button fx:id="settingsButton" mnemonicParsing="false"
onAction="#settingsAction">
    <graphic>
        <ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/settings-icon.png" />
            </image>
        </ImageView>
    </graphic>
    <tooltip>
        <Tooltip text="Settings" />
    </tooltip>
</Button>
<Button fx:id="redoButton1" mnemonicParsing="false"
onAction="#helpAction">
    <graphic>
        <ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/help-icon.png" />
            </image>
        </ImageView>
    </graphic>
    <tooltip>
        <Tooltip text="Help" />
    </tooltip>
</Button>
</items>
</ToolBar>
<SplitPane dividerPositions="0.33" minHeight="490.0" minWidth="800.0"
nodeOrientation="LEFT_TO_RIGHT" VBox.vgrow="ALWAYS">
    <items>
        <VBox minHeight="490.0" minWidth="200.0">
            <children>
                <ScrollPane minHeight="200.0" minWidth="200.0">
                    <content>
                        <TreeView fx:id="programTree" minHeight="600.0"
minWidth="200.0" onDragDetected="#onProgramDragDetected" />
                    </content>
                </ScrollPane>
                <ScrollPane minHeight="289.0" minWidth="200.0">
                    <content>
                        <TreeView fx:id="instructionTree" minHeight="600.0"
minWidth="200.0" onDragDetected="#onInstructionDragDetected" />
                    </content>
```

```
        </ScrollPane>
    </children>
</VBox>
<VBox minHeight="490.0" minWidth="200.0">
    <children>
        <ScrollPane fx:id="ladderPane" minHeight="390.0"
minWidth="200.0" />
        <TextArea fx:id="status" editable="false" minHeight="200.0"
minWidth="200.0" wrapText="true">
            <font>
                <Font size="14.0" />
            </font></TextArea>
        </children>
    </VBox>
</items>
<VBox.margin>
    <Insets />
</VBox.margin>
</SplitPane>
</children>
</VBox>

```

RungView.fxml code

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.*?>
<?import javafx.scene.text.*?>
<?import jav<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<VBox maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity" prefHeight="600.0" prefWidth="800.0"
stylesheets="@css/mainwindow.css" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.vts.jplceditor.ui.controller.MainWindowController">
    <children>
        <MenuBar>
            <menus>
                <Menu mnemonicParsing="false" text="File">
                    <items>
                        <MenuItem mnemonicParsing="false"
onAction="#createNewProject" text="New Project" />
                        <MenuItem mnemonicParsing="false" onAction="#openProject"
text="Open Project" />
                        <RadioMenuItem mnemonicParsing="false"
onAction="#saveProject" text="Save Project" />
                        <MenuItem mnemonicParsing="false" onAction="#onExit"
text="Exit" />
                    </items>
                </Menu>
                <Menu mnemonicParsing="false" text="Edit">
```

```

        <items>
            <MenuItem mnemonicParsing="false" onAction="#undoAction"
text="Undo" />
            <MenuItem mnemonicParsing="false" onAction="#redoAction"
text="Redo" />
            <MenuItem mnemonicParsing="false" onAction="#deleteAction"
text="Delete" />
        </items>
    </Menu>
    <Menu mnemonicParsing="false" text="Compiler">
        <items>
            <MenuItem mnemonicParsing="false" onAction="#verifyProgram"
text="Verify" />
            <MenuItem mnemonicParsing="false"
onAction="#compileProgram" text="Compile" />
        </items>
    </Menu>
    <Menu mnemonicParsing="false" text="Program">
        <items>
            <MenuItem mnemonicParsing="false" onAction="#saveProgram"
text="Save Program" />
            <Menu mnemonicParsing="false" text="Create">
                <items>
                    <MenuItem mnemonicParsing="false" onAction="#createTag"
text="Tag" />
                    <MenuItem mnemonicParsing="false"
onAction="#createRung" text="Rung" />
                    <MenuItem mnemonicParsing="false"
onAction="#createNewLadder" text="Program" />
                </items>
            </Menu>
        </items>
    </Menu>
    <Menu mnemonicParsing="false" text="Communications">
        <items>
            <MenuItem mnemonicParsing="false" onAction="#downloadProgram"
text="Download" />
            <MenuItem mnemonicParsing="false"
onAction="#commSettingsAction" text="Settings" />
        </items>
    </Menu>
    <Menu mnemonicParsing="false" text="Help">
        <items>
            <MenuItem mnemonicParsing="false"
onAction="#documentationAction" text="Documentation" />
        </items>
    </Menu>
</menus>
</MenuBar>
<ToolBar prefHeight="80.0">
    <items>
        <Button fx:id="newLadderButton" mnemonicParsing="false"
onAction="#createNewProject">
            <graphic>
                <Imageview fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
                    <image>

```

```
        <Image url="@../icon/new_ladder-icon.png" />
    </image>
</ImageView>
</graphic>
<tooltip>
    <Tooltip text="New Ladder Program" />
</tooltip>
</Button>
<Button fx:id="saveButton" mnemonicParsing="false"
onAction="#saveProject">
    <graphic>
        <ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/save-icon.png" />
            </image>
        </ImageView>
    </graphic>
    <tooltip>
        <Tooltip text="Save Ladder Program" />
    </tooltip>
</Button>
<Button fx:id="undoButton" mnemonicParsing="false"
onAction="#undoAction">
    <graphic>
        <ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/undo-icon.png" />
            </image>
        </ImageView>
    </graphic>
    <tooltip>
        <Tooltip text="Undo" />
    </tooltip>
</Button>
<Button fx:id="redoButton" mnemonicParsing="false"
onAction="#redoAction">
    <graphic>
        <ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/redo-icon.png" />
            </image>
        </ImageView>
    </graphic>
    <tooltip>
        <Tooltip text="Redo" />
    </tooltip>
</Button>
<Button fx:id="verifyButton" mnemonicParsing="false"
onAction="#verifyProgram">
    <graphic>
        <ImageView fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/ok-apply-icon.png" />
```

```
        </image>
    </Image View>
</graphic>
<tooltip>
    <Tooltip text="Verify Program" />
</tooltip>
</Button>
<Button fx:id="downloadButton" mnemonicParsing="false"
onAction="#downloadProgram">
    <graphic>
        <Image View fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/download-icon.png" />
            </image>
        </Image View>
    </graphic>
    <tooltip>
        <Tooltip text="Download Program" />
    </tooltip>
</Button>
<Button fx:id="settingsButton" mnemonicParsing="false"
onAction="#settingsAction">
    <graphic>
        <Image View fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/settings-icon.png" />
            </image>
        </Image View>
    </graphic>
    <tooltip>
        <Tooltip text="Settings" />
    </tooltip>
</Button>
<Button fx:id="redoButton1" mnemonicParsing="false"
onAction="#helpAction">
    <graphic>
        <Image View fitHeight="64.0" fitWidth="64.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../icon/help-icon.png" />
            </image>
        </Image View>
    </graphic>
    <tooltip>
        <Tooltip text="Help" />
    </tooltip>
</Button>
</items>
</ToolBar>
<SplitPane dividerPositions="0.33" minHeight="490.0" minWidth="800.0"
nodeOrientation="LEFT_TO_RIGHT" VBox.vgrow="ALWAYS">
    <items>
        <VBox minHeight="490.0" minWidth="200.0200.0" minWidth="200.0
```

```

        <content>
            <TreeView fx:id="programTree" minHeight="600.0"
minWidth="200.0" onDragDetected="#onProgramDragDetected" />
        </content>
    </ScrollPane>
    <ScrollPane minHeight="289.0" minWidth="200.0">
        <content>
            <TreeView fx:id="instructionTree" minHeight="600.0"
minWidth="200.0" onDragDetected="#onInstructionDragDetected" />
        </content>
    </ScrollPane>
</children>
</VBox>
<VBox minHeight="490.0" minWidth="200.0">
    <children>
        <ScrollPane fx:id="ladderPane" minHeight="390.0"
minWidth="200.0" />
        <TextArea fx:id="status" editable="false" minHeight="200.0"
minWidth="200.0" wrapText="true">
            <font>
                <Font size="14.0" />
            </font></TextArea>
        </children>
    </VBox>
</items>
<VBox.margin>
    <Insets />
</VBox.margin>
</SplitPane>
</children>
</VBox>


```

Project.fxml code

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.image.*?>
<?import javafx.scene.text.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<AnchorPane id="AnchorPane" prefHeight="300.0" prefWidth="400.0"
xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.vts.jplceditor.ui.controller.ProjectController">
    <children>
        <Label layoutX="21.0" layoutY="43.0" text="Create New JPLC Project">
            <font>
                <Font name="System Bold" size="18.0" />
```

```
        </font>
    </Label>
    <Label layoutX="21.0" layoutY="110.0" text="Project Name" />
    <Label layoutX="21.0" layoutY="159.0" text="Project Directory" />
    <TextField fx:id="nameTF" layoutX="126.0" layoutY="106.0"
prefHeight="25.0" prefWidth="192.0" />
    <Button fx:id="browseButton" layoutX="331.0" layoutY="155.0"
mnemonicParsing="false" onAction="#browseButtonAction" text="Browse" />
    <TextField fx:id="directoryTF" layoutX="126.0" layoutY="155.0"
prefHeight="25.0" prefWidth="192.0" />
    <Button fx:id="cancelButton" layoutX="291.0" layoutY="231.0"
mnemonicParsing="false" onAction="#cancelButtonAction" text="Cancel">
        <graphic>
            <ImageView fitHeight="24.0" fitWidth="24.0" pickOnBounds="true"
preserveRatio="true">
                <image>
                    <Image url="@../icon/cancel-icon.png" />
                </image>
            </ImageView>
        </graphic>
    </Button>
    <Button fx:id="acceptButton" layoutX="172.0" layoutY="231.0"
mnemonicParsing="false" onAction="#acceptButtonAction" text="Accept">
        <graphic>
            <ImageView fitHeight="24.0" fitWidth="24.0" pickOnBounds="true"
preserveRatio="true">
                <image>
                    <Image url="@../icon/ok-apply-icon.png" />
                </image>
            </ImageView>
        </graphic>
    </Button>
</children>
</AnchorPane>
```

Java Classes

```
package com.vts.jplceditor.ui.controller;

import com.vts.jplceditor.resource.R;
import com.vts.jplceditor.compiler.ld.LDCompiler;
import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.ld.LadderDiagram;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import com.vts.jplceditor.ld.instruction.InstructionUtil;
import com.vts.jplceditor.project.JPLCProject;
import com.vts.jplceditor.project.SavableProject;
import java.io.File;
import java.io.IOException;
import java.net.URL;
import java.util.HashMap;
import java.util.ResourceBundle;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.application.Platform;
import javafx.beans.property.ObjectProperty;
```

```
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.TreeItem;
import javafx.scene.control.TreeView;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.input.ClipboardContent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.MouseEvent;
import javafx.scene.input.TransferMode;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import com.vts.jplceditor.ld.instruction.Instruction;
import javafx.scene.control.TextArea;

/**
 *
 * @author Vusumuzi_Tshabangu
 */
public class MainWindowController implements Initializable {

    @FXML
    private Button newLadderButton;
    @FXML
    private Button saveButton;
    @FXML
    private Button undoButton;
    @FXML
    private Button redoButton;
    @FXML
    private Button verifyButton;
    @FXML
    private Button downloadButton;
    @FXML
    private Button settingsButton;
    @FXML
    private Button redoButton1;
    @FXML
    private ScrollPane ladderPane;
    @FXML
    private TreeView<String> programTree;
    @FXML
    private TreeView<String> instructionTree;
    @FXML
    private TextArea status;

    private Stage popupStage;
    private URL url = null;
    private Parent root = null;
    private HashMap<String, String> tags;
    private TreeItem<String> tagList;
```

```
private TreeItem<String> ioList;
private TreeItem<String> timerList;
private TreeItem<String> ladderList;
private TreeItem<String> instructionList;
private TreeItem<String> programTreeRoot;
private Dragboard dragBoard;
private ClipboardContent dragContent;
private JPLCProject project;
private LadderDiagram ladderDiagram;
private LDCompiler compiler;

@Override
public void initialize(URL url, ResourceBundle rb) {

    ImageView rootIcon = new ImageView(new Image(R.ICON_FOLDER));
    programTreeRoot = new TreeItem<>("Ladder Program", rootIcon);
    programTree.setRoot(programTreeRoot);

    ImageView instructionRoot = new ImageView(new Image(R.ICON_FOLDER));
    ImageView tagRoot = new ImageView(new Image(R.ICON_TAG_FOLDER));
    Image programRoot = new Image(R.ICON_LADDER_FOLDER);

    tagList = new TreeItem<>("Tags", tagRoot);
    ioList = UIUtil.buildIOList();
    timerList = UIUtil.buildTimerList();
    ladderList = new TreeItem<>("Programs", new ImageView(programRoot));
    instructionList = UIUtil.buildInstructionList();

    programTreeRoot.getChildren().add(ladderList);
    programTreeRoot.getChildren().add(tagList);
    programTreeRoot.getChildren().add(ioList);
    programTreeRoot.getChildren().add(timerList);
    programTreeRoot.setExpanded(true);

    instructionTree.setRoot(instructionList);
    instructionList.setExpanded(true);
    popupStage = new Stage();

    tags = new HashMap<>();
    UIUtil.buildIoList(tags);
}

@FXML
private void saveProgram(ActionEvent event) {
}

@FXML
private void onExit(ActionEvent event) {
    System.exit(0);
}

@FXML
private void redoAction(ActionEvent event) {
}

@FXML
private void verifyProgram(ActionEvent event) {
```

```
}

@FXML
private void createTag(ActionEvent event) {
    url = getClass().getResource(R.WINDOW_TAG);
    showPopup(Popup.TagWindow);
}

@FXML
private void createRung(ActionEvent event) {
    url = getClass().getResource(R.WINDOW_RUNG);
    showPopup(Popup.RungWindow);
}

@FXML
private void undoAction(ActionEvent event) {
}

@FXML
private void downloadProgram(ActionEvent event) {
    if (project != null) {
        status.appendText(project.download().toString());
    } else {
        status.appendText("\nYou need to create a Project and Compile it
First!!!!!!\n");
    }
}

@FXML
private void settingsAction(ActionEvent event) {
}

@FXML
private void helpAction(ActionEvent event) {
}

@FXML
private void createNewProject(ActionEvent event) {
    url = getClass().getResource(R.WINDOW_PROJECT);
    showPopup(Popup.ProjectWindow);
}

@FXML
private void openProject(ActionEvent event) {
    FileChooser chooser = new FileChooser();
    final File projFile = chooser.showOpenDialog(null);
    if (projFile != null) {
        SavableProject pro = JPLCProject.readInProject(projFile);
        project = new JPLCProject(pro);
        ladderDiagram = project.getDefaultLD();
        ladderPane.setContent(ladderDiagram);
    }
}

@FXML
private void saveProject(ActionEvent event) {
    if (project != null) {
```

```
        try {
            status.appendText(project.saveProject().toString());
        } catch (IOException ex) {

Logger.getLogger(MainWindowController.class.getName()).log(Level.SEVERE,
null, ex);
    }
}

@FXML
private void deleteAction(ActionEvent event) {

}

@FXML
private void compileProgram(ActionEvent event) {
    if (project != null) {
        status.clear();
        status.appendText(project.compileProject().toString());
    } else {
        status.appendText("\nYou need to create a Project First
!!!!!!!!!!!!\n");
    }
}

@FXML
private void commSettingsAction(ActionEvent event) {

}

@FXML
private void documentationAction(ActionEvent event) {

}

@FXML
private void createNewLadder(ActionEvent event) {
    url = getClass().getResource(R.WINDOW_PROGRAM);
    showPopup(Popup.ProgramWindow);
}

@FXML
private void onProgramDragDetected(MouseEvent event) {
    dragBoard = programTree.startDragAndDrop(TransferMode.COPY);
    dragContent = new ClipboardContent();
    String selected =
programTree.getSelectionModel().getSelectedItem().getValue();
    String parent =
programTree.getSelectionModel().getSelectedItem().getParent().getValue();
    boolean isTag = parent.equals("Tags") || parent.equals("Digital
Inputs")
        || parent.equals("Digital Outputs") || parent.equals("Analog
Inputs") || parent.equals("Analog Outputs");
    if (isTag) {
        String type = tags.get(selected);
        String dragData = DragDataFormatter.formatTag(selected, type);
        dragContent.putString(dragData);
        dragBoard.setContent(dragContent);
    }
}
```

```

        if (parent.equals("Timer") || parent.equals("Counter")) {
            String type;
            if (selected.contains("TT") || selected.contains("DN")) {
                type = DragDataFormatter.TagType_BIT;
            } else {
                type = DragDataFormatter.TagType_INT;
            }
            String dragData = DragDataFormatter.formatTag(selected, type);
            dragContent.putString(dragData);
            dragBoard.setContent(dragContent);
        }

        event.consume();
    }

    @FXML
    private void onInstructionDragDetected(MouseEvent event) {

        String selected =
instructionTree.getSelectionModel().getSelectedItem().getValue();
        String parent =
instructionTree.getSelectionModel().getSelectedItem().getParent().getValue();
        Instruction i = InstructionUtil.getInstruction(selected);
        if (i != null) {
            dragBoard = instructionTree.startDragAndDrop(TransferMode.COPY);
            dragContent = new ClipboardContent();
            dragContent.putString(selected);
            dragBoard.setContent(dragContent);
        } else {
            System.out.println("Null Instruction:" + selected);
        }
        event.consume();
    }

    private void showPopup(Popup popup) {
        if (url != null) {
            try {

                FXMLLoader load = new FXMLLoader(url);
                root = load.load();
                if (popup == Popup.TagWindow) {
                    TagWindowController con = (TagWindowController)
load.getController();
                    if (con != null) {
                        con.setController(this);
                    }
                } else if (popup == Popup.RungWindow) {
                    RungViewController con = (RungViewController)
load.getController();
                    if (con != null) {
                        con.setController(this);
                    }
                } else if (popup == Popup.ProgramWindow) {
                    ProgramViewController con = (ProgramViewController)
load.getController();
                    if (con != null) {
                        con.setController(this);
                    }
                }
            }
        }
    }
}

```

```

        }
    } else if (popup == Popup.ProjectWindow) {
        ProjectController con = (ProjectController)
load.getController();
        if (con != null) {
            con.setController(this);
        }
    }
    popupStage.setScene((new Scene(root)));
    popupStage.show();
} catch (IOException ex) {

Logger.getLogger(MainWindowController.class.getName()).log(Level.SEVERE,
null, ex);
}
}

/**
 *
 */
public void closePopup() {
    popupStage.close();
}

/**
 *
 * @param tagName
 * @param tagType
 */
public void addTag(String tagName, String tagType) {
    ImageView tagIcon = new ImageView(new Image(R.ICON_TAG_ICON));
    tagList.getChildren().add(new TreeItem<>(tagName, tagIcon));
    tags.put(tagName, tagType);
    PLCDataType type = UIUtil.typeFromString(tagType);
    project.addTag(type, tagName);
    closePopup();
}

/**
 *
 * @param rungNumber
 */
public void addRung(int rungNumber) {
    ladderDiagram.addRung(rungNumber);
    closePopup();
}

/**
 *
 * @param programName
 */
public void addProgram(String programName) {
    ImageView rungIcon = new ImageView(new Image(R.ICON_LADDER_ICON));
    ladderList.getChildren().add(new TreeItem<>(programName, rungIcon));
    closePopup();
}
}

```

```
/**
 *
 * @param name
 * @param directory
 */
public void createProject(String name, String directory) {
    project = new JPLCProject(directory, name);
    ladderDiagram = project.getDefaultLD();
    ladderPane.setContent(ladderDiagram);
    closePopup();
}

/**
 * Generic method for putting element running on a non-JavaFX thread on
the
 * JavaFX thread, to properly update the UI
 *
 * @param property a {@link ObjectProperty}
 * @param value the value to set for the given {@link ObjectProperty}
 */
private <T> void onFXThread(final ObjectProperty<T> property, final T
value) {
    Platform.runLater(new Runnable() {

        @Override
        public void run() {
            property.set(value);
        }
    });
}

}

package com.vts.jplceditor.ui.controller;

public enum Popup {

    TagWindow,      */
    RungWindow,
    ProgramWindow,
    ProjectWindow
}

package com.vts.jplceditor.ui.controller;
```

```
import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import com.vts.jplceditor.resource.R;
import java.util.HashMap;
import javafx.scene.control.TreeItem;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;

/**
 *
 * @author Vusumuzi_Tshabangu
 */
public class UIUtil {

    /**
     *
     * @return
     */
    public static TreeItem<String> buildInstructionList() {
        Image rootIcon = new Image(R.ICON_LADDER_FOLDER);
        Image elementIcon = new Image(R.ICON_INSTR_ICON);
        TreeItem<String> root = new TreeItem<>("Instructions", new
        ImageView(rootIcon));
        TreeItem<String> relay = new TreeItem<>("Relay", new
        ImageView(rootIcon));
        for (String name : R.LIST_INSTR_RELAY) {
            relay.getChildren().add(new TreeItem<>(name, new
            ImageView(elementIcon)));
        }
        TreeItem<String> timer = new TreeItem<>("Timer", new
        ImageView(rootIcon));
        for (String name : R.LIST_INSTR_TIMER) {
            timer.getChildren().add(new TreeItem<>(name, new
            ImageView(elementIcon)));
        }

        TreeItem<String> compare = new TreeItem<>("Compare", new
        ImageView(rootIcon));
        for (String name : R.LIST_INSTR_COMPARE) {
            compare.getChildren().add(new TreeItem<>(name, new
            ImageView(elementIcon)));
        }
        TreeItem<String> math = new TreeItem<>("Math", new
        ImageView(rootIcon));
        for (String name : R.LIST_INSTR_MATH) {
            math.getChildren().add(new TreeItem<>(name, new
            ImageView(elementIcon)));
        }

        TreeItem<String> logical = new TreeItem<>("Logical", new
        ImageView(rootIcon));
        for (String name : R.LIST_INSTR_LOGICAL) {
            logical.getChildren().add(new TreeItem<>(name, new
            ImageView(elementIcon)));
        }
    }
}
```

```
        TreeItem<String> file = new TreeItem<>("File", new
ImageView(rootIcon));
        for (String name : R.LIST_INSTR_FILE) {
            file.getChildren().add(new TreeItem<>(name, new
ImageView(elementIcon)));
        }

        root.getChildren().add(relay);
        root.getChildren().add(timer);
        root.getChildren().add(math);
        root.getChildren().add(compare);
        root.getChildren().add(file);
        root.getChildren().add(logical);

    return root;
}

/**
 *
 * @return
 */
public static TreeItem<String> buildTimerList() {
    Image rootIcon = new Image(R.ICON_TAG_FOLDER);
    Image elementIcon = new Image(R.ICON_TAG_ICON);
    TreeItem<String> root = new TreeItem<>("Timers", new
ImageView(rootIcon));

    TreeItem<String> timer = new TreeItem<>("Timer", new
ImageView(rootIcon));
    for (String name : R.LIST_DATA_TIMER) {
        timer.getChildren().add(new TreeItem<>(name, new
ImageView(elementIcon)));
    }

    TreeItem<String> counter = new TreeItem<>("Counter", new
ImageView(rootIcon));
    for (String name : R.LIST_DATA_COUNTER) {
        counter.getChildren().add(new TreeItem<>(name, new
ImageView(elementIcon)));
    }

    root.getChildren().add(timer);
    root.getChildren().add(counter);

    return root;
}

/**
 *
 * @return
 */
public static TreeItem<String> buildIOLIST() {
    Image rootIcon = new Image(R.ICON_TAG_FOLDER);
    Image diiIcon = new Image(R.ICON_TAG_FOLDER);
    Image dioIcon = new Image(R.ICON_TAG_FOLDER);
    Image anInIcon = new Image(R.ICON_TAG_FOLDER);
    Image anOutIcon = new Image(R.ICON_TAG_FOLDER);
```

```

        Image elementIcon = new Image(R.ICON_TAG_ICON);

        TreeItem<String> root = new TreeItem<>("I/O", new
        ImageView(rootIcon));
        TreeItem<String> diiItems = new TreeItem<>("Digital Inputs", new
        ImageView(rootIcon));
        TreeItem<String> dioItems = new TreeItem<>("Digital Outputs", new
        ImageView(rootIcon));
        TreeItem<String> anInItems = new TreeItem<>("Analog Inputs", new
        ImageView(rootIcon));
        TreeItem<String> anOutItems = new TreeItem<>("Analog Outputs", new
        ImageView(rootIcon));

        String[] diodata = new String[]{
            "DO0", "DO1", "DO2", "DO3", "DO4", "DO5", "DO6", "DO7"
        };
        String[] diidata = new String[]{
            "DI0", "DI1", "DI2", "DI3", "DI4", "DI5", "DI6", "DI7"
        };
        String[] anIndata = new String[]{
            "AI0", "AI1", "AI2", "AI3", "AI4", "AI5", "AI6", "AI7",
        };
        String[] anOutdata = new String[]{
            "AO0"
        };
        for (String name : diidata) {
            diiItems.getChildren().add(new TreeItem<>(name, new
        ImageView(elementIcon)));
        }
        for (String name : diodata) {
            dioItems.getChildren().add(new TreeItem<>(name, new
        ImageView(elementIcon)));
        }
        for (String name : anIndata) {
            anInItems.getChildren().add(new TreeItem<>(name, new
        ImageView(elementIcon)));
        }
        for (String name : anOutdata) {
            anOutItems.getChildren().add(new TreeItem<>(name, new
        ImageView(elementIcon)));
        }
        root.getChildren().add(diiItems);
        root.getChildren().add(dioItems);
        root.getChildren().add(anInItems);
        root.getChildren().add(anOutItems);
        return root;
    }

    /**
     *
     * @param type
     * @return
     */
    public static PLCDATAType typeFromString(String type) {
        if (type.equals(DragDataFormatter.TagType_BIT)) {
            return PLCDATAType.BIT;
        } else if (type.equals(DragDataFormatter.TagType_BYTE)) {

```

```

        return PLCDataType.BYTE;
    } else if (type.equals(DragDataFormatter.TagType_FLOAT)) {
        return PLCDataType.FLOAT;
    } else if (type.equals(DragDataFormatter.TagType_INT)) {
        return PLCDataType.INT;
    } else {
        return null;
    }
}

public static void buildIoList(HashMap<String, String> tags) {
    tags.put("DI0", DragDataFormatter.TagType_BIT);
    tags.put("DI1", DragDataFormatter.TagType_BIT);
    tags.put("DI2", DragDataFormatter.TagType_BIT);
    tags.put("DI3", DragDataFormatter.TagType_BIT);
    tags.put("DI4", DragDataFormatter.TagType_BIT);
    tags.put("DI5", DragDataFormatter.TagType_BIT);
    tags.put("DI6", DragDataFormatter.TagType_BIT);
    tags.put("DI7", DragDataFormatter.TagType_BIT);

    tags.put("DO0", DragDataFormatter.TagType_BIT);
    tags.put("DO1", DragDataFormatter.TagType_BIT);
    tags.put("DO2", DragDataFormatter.TagType_BIT);
    tags.put("DO3", DragDataFormatter.TagType_BIT);
    tags.put("DO4", DragDataFormatter.TagType_BIT);
    tags.put("DO5", DragDataFormatter.TagType_BIT);
    tags.put("DO6", DragDataFormatter.TagType_BIT);
    tags.put("DO7", DragDataFormatter.TagType_BIT);

    tags.put("AI0", DragDataFormatter.TagType_INT);
    tags.put("AI1", DragDataFormatter.TagType_INT);
    tags.put("AI2", DragDataFormatter.TagType_INT);
    tags.put("AI3", DragDataFormatter.TagType_INT);
    tags.put("AI4", DragDataFormatter.TagType_INT);
    tags.put("AI5", DragDataFormatter.TagType_INT);
    tags.put("AI6", DragDataFormatter.TagType_INT);
    tags.put("AI7", DragDataFormatter.TagType_INT);
    tags.put("AO0", DragDataFormatter.TagType_INT);
}
}

package com.vts.jplceditor.ui.controller;

import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;

/**
 * FXML Controller class
 *
 * @author Vusivus
 */
public class ProgramViewController implements Initializable {

```

```
@FXML
private Button acceptButton;
@FXML
private Button cancelButton;
@FXML
private TextField nameTF;

private MainWindowController controller;

/**
 * Initializes the controller class.
 * @param controller
 */
public void setController(MainWindowController controller) {
    this.controller = controller;
}

@Override
public void initialize(URL url, ResourceBundle rb) {
    // TODO
}

@FXML
private void acceptButtonAction(ActionEvent event) {
    String name = nameTF.getText();
    controller.addProgram(name);
}

@FXML
private void cancelButtonAction(ActionEvent event) {
    controller.closePopup();
}

}

package com.vts.jplceditor.ui.controller;

import java.io.File;
import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.stage.DirectoryChooser;

/**
 * FXML Controller class
 *
 * @author Vusivus
 */
```

```
public class ProjectController implements Initializable {

    @FXML
    private TextField nameTF;
    @FXML
    private Button browseButton;
    @FXML
    private TextField directoryTF;
    private MainWindowController controller;
    @FXML
    private Button cancelButton;
    @FXML
    private Button acceptButton;

    /**
     * Initializes the controller class.
     * @param url
     * @param rb
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }

    @FXML
    private void browseButtonAction(ActionEvent event) {
        DirectoryChooser chooser = new DirectoryChooser();
        final File directory = chooser.showDialog(null);
        if (directory != null) {
            directoryTF.setText(directory.getAbsolutePath()+"\\");
        }
    }

    void setController(MainWindowController con) {
        controller = con;
    }

    @FXML
    private void cancelButtonAction(ActionEvent event) {
        controller.closePopup();
    }

    @FXML
    private void acceptButtonAction(ActionEvent event) {
        String name, dir;
        name = nameTF.getText();
        dir = directoryTF.getText();

        if (!"".equals(name) && !"".equals(dir)) {
            controller.createProject(name, dir);
        }
    }
}

package com.vts.jplceditor.ui.controller;
```

```
import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;

/**
 * FXML Controller class
 *
 * @author Vusumuzi_Tshabangu
 */
public class RungViewController implements Initializable {

    @FXML
    private Button acceptButton;
    @FXML
    private Button cancelButton;
    @FXML
    private TextField numberTF;

    private MainWindowController controller;

    /**
     * Initializes the controller class.
     *
     * @param url
     * @param rb
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }

    /**
     * @param controller
     */
    public void setController(MainWindowController controller) {
        this.controller = controller;
    }

    @FXML
    private void acceptButtonAction(ActionEvent event) {
        if (isInteger(numberTF.getText())) {
            int number = Integer.valueOf(numberTF.getText());
            controller.addRung(number);
        }
    }

    @FXML
    private void cancelButtonAction(ActionEvent event) {
        controller.closePopup();
    }
}
```

```
public static boolean isInteger(String string) {  
    if (string.isEmpty()) {  
        return false;  
    }  
    for (int i = 0; i < string.length(); i++) {  
        if (i == 0 && string.charAt(i) == '-') {  
            if (string.length() == 1) {  
                return false;  
            } else {  
                continue;  
            }  
        }  
        if (Character.digit(string.charAt(i), 10) < 0) {  
            return false;  
        }  
    }  
    return true;  
}  
}
```

```
package com.vts.jplceditor.ui.controller;  
  
import java.net.URL;  
import java.util.ResourceBundle;  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.fxml.Initializable;  
import javafx.scene.control.Button;  
import javafx.scene.control.TextField;  
  
/**  
 * FXML Controller class  
 *  
 * @author Vusumuzi_Tshabangu  
 */  
public class RungViewController implements Initializable {  
  
    @FXML  
    private Button acceptButton;  
    @FXML  
    private Button cancelButton;  
    @FXML  
    private TextField numberTF;  
  
    private MainWindowController controller;  
  
    /**  
     * Initializes the controller class.  
     *  
     * @param url  
     * @param rb  
     */
```

```
@Override
public void initialize(URL url, ResourceBundle rb) {
    // TODO
}

/**
 *
 * @param controller
 */
public void setController(MainWindowController controller) {
    this.controller = controller;
}

@FXML
private void acceptButtonAction(ActionEvent event) {
    if (isInteger(numberTF.getText())) {
        int number = Integer.valueOf(numberTF.getText());
        controller.addRung(number);
    }
}

@FXML
private void cancelButtonAction(ActionEvent event) {
    controller.closePopup();
}

public static boolean isInteger(String string) {
    if (string.isEmpty()) {
        return false;
    }
    for (int i = 0; i < string.length(); i++) {
        if (i == 0 && string.charAt(i) == '-') {
            if (string.length() == 1) {
                return false;
            } else {
                continue;
            }
        }
        if (Character.digit(string.charAt(i), 10) < 0) {
            return false;
        }
    }
    return true;
}
}
```

Appendix F: JPLC Editor resource layer classes code

```
package com.vts.jplceditor.resource;

/**
 *
 * @author Vusumuzi_Tshabangu
 */
public class R {

    /**
     *
     */
    public static String WINDOW_MAIN =
"/com/vts/jplceditor/ui/fxml/MainWindow.fxml";

    /**
     *
     */
    public static String WINDOW_PROJECT =
"/com/vts/jplceditor/ui/fxml/Project.fxml";

    /**
     *
     */
    public static String WINDOW_TAG =
"/com/vts/jplceditor/ui/fxml/TagWindow.fxml";

    /**
     *
     */
    public static String WINDOW_RUNG =
"/com/vts/jplceditor/ui/fxml/RungView.fxml";

    /**
     *
     */
    public static String WINDOW_PROGRAM =
"/com/vts/jplceditor/ui/fxml/ProgramView.fxml";

    /**
     *
     */
    public static String OBJECT_CONNECTION =
"/com/vts/jplceditor/ui/ladder/Connection.fxml";

    /**
     *
     */
    public static String OBJECT_RAIL =
"/com/vts/jplceditor/ui/ladder/Rail.fxml";

    /**
     *
     */
}
```

```
    public static String ICON_TAG_FOLDER =
"/com/vts/jplceditor/resource/icon/tag_folder.png";

    /**
     *
     */
    public static String ICON_INSTR_ICON =
"/com/vts/jplceditor/resource/icon/pencil-icon.png";

    /**
     *
     */
    public static String ICON_TAG_ICON =
"/com/vts/jplceditor/resource/icon/tag_icon.png";

    /**
     *
     */
    public static String ICON_LADDER_ICON =
"/com/vts/jplceditor/resource/icon/ladder_icon.png";

    /**
     *
     */
    public static String ICON_LADDER_FOLDER =
"/com/vts/jplceditor/resource/icon/program_folder.png";

    /**
     *
     */
    public static String ICON_FOLDER =
"/com/vts/jplceditor/resource/icon/folder-icon.png";

    /**
     *
     */
    public static String[] LIST_INSTR_RELAY = new String[]{
        "BRN", "XIC", "XIO", "OTE", "OTL", "OTU",
    };

    /**
     *
     */
    public static String[] LIST_INSTR_TIMER = new String[]{
        "TON", "CTU", "CTD"
    };

    /**
     *
     */
    public static String[] LIST_INSTR_COMPARE = new String[]{
        "EQU", "GRT", "LES", "LIM"
    };

    /**
     *
     */
```

```
public static String[] LIST_INSTR_MATH = new String[]{  
    "ADD", "SUB", "MUL", "DIV", "SQR"  
};  
  
/**  
 *  
 */  
public static String[] LIST_INSTR_LOGICAL = new String[]{  
    "AND", "OR", "XOR",  
};  
  
/**  
 *  
 */  
public static String[] LIST_INSTR_FILE = new String[]{  
    "MOV", "MVM"  
};  
  
/**  
 *  
 */  
public static String[] LIST_DATA_TIMER = new String[]{  
    "T-DN", "T-TT", "T-COUNT"  
};  
  
/**  
 *  
 */  
public static String[] LIST_DATA_COUNTER = new String[]{  
    "C-DN", "C-COUNT"  
};  
};
```

Appendix G: JPLC Editor Project layer classes code

```

package com.vts.jplceditor.project.io;

import com.vts.jplceditor.ld.LadderDiagram;
import com.vts.jplceditor.ld.component.LDInstructionManager;
import com.vts.jplceditor.ld.component.LDRung;
import com.vts.jplceditor.ld.component.LDInstruction;
import com.vts.jplceditor.ld.component.node.SNode;
import java.util.ArrayList;
import java.util.List;
import javafx.geometry.Point2D;

/**
 *
 * @author Vusivus
 */
public class LDParser {

    /**
     * @param ld
     * @return
     */
    public static IOULD fromLadderDiagram(LadderDiagram ld) {
        IOULD ioladder = new IOULD();
        //Create IONode list
        List<SNode> sNodes = ld.getSourceNodes();
        List<IONode> nodes = new ArrayList<>();
        sNodes.stream().forEach((sNode) -> {
            nodes.add(fromSNode(sNode));
        });
        ioladder.setsNodes(nodes);
        //Create IORungList
        List<LDRung> rungs = ld.getRungList();
        List<IORung> rList = new ArrayList<>();
        rungs.stream().map((rung) -> {
            IORung ior = new IORung(fromSNode(rung.getsNode()));
            ior.setSize(rung.getSize());
            List<LDInstruction> inodes = rung.getInstructionList();
            List<IOInstruction> iList = new ArrayList<>();
            inodes.stream().map((inode) -> {
                IOInstruction i = new
IOInstruction(inode.getStartPoint().getX(), inode.getStartPoint().getY());
                i.setSize(inode.getSize());
                //    i.setInstruction(inode.getInstruction());
                //    i.setParameters(inode.getparameters());
                return i;
            }).forEach((i) -> {
                iList.add(i);
            });
            ior.setInstructions(iList);
            return ior;
        }).forEach((ior) -> {
            rList.add(ior);
        });
    }
}

```

```

        ioladder.setRungs(rList);
        return ioladder;
    }

    private static IONode fromSNode(SNode sNode) {
        return new IONode(sNode.getNodeType(), sNode.getCenterX(),
sNode.getCenterY());
    }

    /**
     *
     * @param ld
     * @return
     */
    public static LadderDiagram fromIOLD(IOLD ld) {
        LadderDiagram ladder;
        //Create SourceNode List
        List<IONode> nodes = ld.getsNodes();
        List<SNode> sourceNodes = new ArrayList<>();
        nodes.stream().forEach((node) -> {
            sourceNodes.add(new SNode(new Point2D(node.getLocationX(),
node.getLocationY())));
        });
        //Create LDRung List
        List<IORung> rungs = ld.getRungs();
        List<LDRung> ldRungs = createRungList(rungs);

        ladder = new LadderDiagram(ldRungs, sourceNodes);
        return ladder;
    }

    private static List<LDRung> createRungList(List<IORung> rungs) {
        List<LDRung> ldRungs = new ArrayList<>();
        for (IORung rung : rungs) {
            List<IOInstruction> ins = rung.getInstructions();
            List<LDInstruction> inodes = new ArrayList<>();
            ins.stream().map((i) -> {
                LDInstruction inode = new LDInstruction(new
Point2D(i.getLocationX(), i.getLocationY()));
                if (i.getInstruction() != null) {
                    inode.setInstruction(i.getInstruction());
                }
                return inode;
            }).forEach((inode) -> {
                inodes.add(inode);
            });
            SNode sn = new SNode(new Point2D(rung.getsNode().getLocationX(),
rung.getsNode().getLocationY()));
            LDInstructionManager manager = new
LDInstructionManager(sn.getLocation(), inodes);

            ldRungs.add(new LDRung(manager, rung.getSize(), sn));
        }
        return ldRungs;
    }
}

```

```
package com.vts.jplceditor.project.io;

import com.vts.jplceditor.ld.LadderDiagram;
import com.vts.jplceditor.ld.component.LDInstructionManager;
import com.vts.jplceditor.ld.component.LDRung;
import com.vts.jplceditor.ld.component.LDInstruction;
import com.vts.jplceditor.ld.component.node.SNode;
import java.util.ArrayList;
import java.util.List;
import javafx.geometry.Point2D;

/**
 *
 * @author Vusivus
 */
public class LDParser {

    /**
     * @param ld
     * @return
     */
    public static IO LD fromLadderDiagram(LadderDiagram ld) {
        IO ioladder = new IO();
        //Create IONode list
        List<SNode> sNodes = ld.getSourceNodes();
        List<IONode> nodes = new ArrayList<>();
        sNodes.stream().forEach((sNode) -> {
            nodes.add(fromSNode(sNode));
        });
        ioladder.setsNodes(nodes);
        //Create IORungList
        List<LDRung> rungs = ld.getRungList();
        List<IORung> rList = new ArrayList<>();
        rungs.stream().map((rung) -> {
            IORung ior = new IORung(fromSNode(rung.getsNode()));
            ior.setSize(rung.getSize());
            List<LDInstruction> inodes = rung.getInstructionList();
            List<IOInstruction> iList = new ArrayList<>();
            inodes.stream().map((inode) -> {
                IOInstruction i = new
                IOInstruction(inode.getStartPoint().getX(), inode.getStartPoint().getY());
                i.setSize(inode.getSize());
                //    i.setInstruction(inode.getInstruction());
                //    i.setParameters(inode.getparameters());
                return i;
            }).forEach((i) -> {
                iList.add(i);
            });
            ior.setInstructions(iList);
            return ior;
        }).forEach((ior) -> {
            rList.add(ior);
        });
        ioladder.setRungs(rList);
        return ioladder;
    }
}
```

```

}

private static IONode fromSNode(SNode sNode) {
    return new IONode(sNode.getNodeType(), sNode.getCenterX(),
sNode.getCenterY());
}

/**
 *
 * @param ld
 * @return
 */
public static LadderDiagram fromIOLD(IOLD ld) {
    LadderDiagram ladder;
    //Create SourceNode List
    List<IONode> nodes = ld.getsNodes();
    List<SNode> sourceNodes = new ArrayList<>();
    nodes.stream().forEach((node) -> {
        sourceNodes.add(new SNode(new Point2D(node.getLocationX(),
node.getLocationY())));
    });
    //Create LDRung List
    List<IORung> rungs = ld.getRungs();
    List<LDRung> ldRungs = createRungList(rungs);

    ladder = new LadderDiagram(ldRungs, sourceNodes);
    return ladder;
}

private static List<LDRung> createRungList(List<IORung> rungs) {
    List<LDRung> ldRungs = new ArrayList<>();
    for (IORung rung : rungs) {
        List<IOInstruction> ins = rung.getInstructions();
        List<LDInstruction> inodes = new ArrayList<>();
        ins.stream().map((i) -> {
            LDInstruction inode = new LDInstruction(new
Point2D(i.getLocationX(), i.getLocationY()));
            if (i.getInstruction() != null) {
                inode.setInstruction(i.getInstruction());
            }
            return inode;
        }).forEach((inode) -> {
            inodes.add(inode);
        });
        SNode sn = new SNode(new Point2D(rung.getsNode().getLocationX(),
rung.getsNode().getLocationY()));
        LDInstructionManager manager = new
LDInstructionManager(sn.getLocation(),inodes);

        ldRungs.add(new LDRung(manager, rung.getSize(), sn));
    }
    return ldRungs;
}

package com.vts.jplceditor.project.io;

```

```
import com.vts.jplceditor.ld.component.node.NodeType;

/**
 *
 * @author Vusivus
 */
public class IONode {
    private NodeType nodeType;
    private double locationX;
    private double locationY;

    /**
     *
     * @param nodeType
     * @param locationX
     * @param locationY
     */
    public IONode(NodeType nodeType, double locationX, double locationY) {
        this.nodeType = nodeType;
        this.locationX = locationX;
        this.locationY = locationY;
    }

    /**
     *
     * @return
     */
    public NodeType getNodeType() {
        return nodeType;
    }

    /**
     *
     * @param nodeType
     */
    public void setNodeType(NodeType nodeType) {
        this.nodeType = nodeType;
    }

    /**
     *
     * @return
     */
    public double getLocationX() {
        return locationX;
    }

    /**
     *
     * @param locationX
     */
    public void setLocationX(double locationX) {
        this.locationX = locationX;
    }

    /**
     *
```

```
    * @return
    */
    public double getLocationY() {
        return locationY;
    }

    /**
     *
     * @param locationY
     */
    public void setLocationY(double locationY) {
        this.locationY = locationY;
    }

}

package com.vts.jplceditor.project.io;

import com.vts.jplceditor.ld.component.node.NodeType;

/**
 *
 * @author Vusivus
 */
public class IONode {
    private NodeType nodeType;
    private double locationX;
    private double locationY;

    /**
     *
     * @param nodeType
     * @param locationX
     * @param locationY
     */
    public IONode(NodeType nodeType, double locationX, double locationY) {
        this.nodeType = nodeType;
        this.locationX = locationX;
        this.locationY = locationY;
    }

    /**
     *
     * @return
     */
    public NodeType getNodeType() {
        return nodeType;
    }

    /**
     *
     * @param nodeType
     */
    public void setNodeType(NodeType nodeType) {
        this.nodeType = nodeType;
    }
}
```

```
/*
 *
 * @return
 */
public double getLocationX() {
    return locationX;
}

/**
 *
 * @param locationX
 */
public void setLocationX(double locationX) {
    this.locationX = locationX;
}

/**
 *
 * @return
 */
public double getLocationY() {
    return locationY;
}

/**
 *
 * @param locationY
 */
public void setLocationY(double locationY) {
    this.locationY = locationY;
}

}

package com.vts.jplceditor.project.io;

import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.compiler.plc.PLCOperator;
import com.vts.jplceditor.ld.layout.Size;
import java.util.List;

/**
 *
 * @author Vusivus
 */
public class IOInstruction {
    private double locationX;
    private double locationY;
    private Size size;
    private PLCOperator instruction;
    private List<PLCOOperand> parameters;

}
```

```
* @param locationX
* @param locationY
*/
public IOInstruction(double locationX, double locationY) {
    this.locationX = locationX;
    this.locationY = locationY;
}

/**
 *
 * @return
 */
public double getLocationX() {
    return locationX;
}

/**
 *
 * @param locationX
 */
public void setLocationX(double locationX) {
    this.locationX = locationX;
}

/**
 *
 * @return
 */
public double getLocationY() {
    return locationY;
}

/**
 *
 * @param locationY
 */
public void setLocationY(double locationY) {
    this.locationY = locationY;
}

/**
 *
 * @return
 */
public Size getSize() {
    return size;
}

/**
 *
 * @param size
 */
public void setSize(Size size) {
    this.size = size;
}

/**
```

```
*  
 * @return  
 */  
 public PLCOperator getInstruction() {  
     return instruction;  
 }  
  
/**  
 *  
 * @param instruction  
 */  
 public void setInstruction(PLCOperator instruction) {  
     this.instruction = instruction;  
 }  
  
/**  
 *  
 * @return  
 */  
 public List<PLCOOperand> getParameters() {  
     return parameters;  
 }  
  
/**  
 *  
 * @param parameters  
 */  
 public void setParameters(List<PLCOOperand> parameters) {  
     this.parameters = parameters;  
 }  
}  
}
```

```
package com.vts.jplceditor.project;  
  
import com.thoughtworks.xstream.XStream;  
import com.vts.jplceditor.comms.usb.SessionManager;  
import com.vts.jplceditor.compiler.ld.LDCompiler;  
import com.vts.jplceditor.compiler.pic18.Pic18Prog;  
import com.vts.jplceditor.compiler.plc.PLCDATAType;  
import com.vts.jplceditor.compiler.plc.PLCOOperand;  
import com.vts.jplceditor.ld.LadderDiagram;  
import com.vts.jplceditor.project.io.IOLD;  
import com.vts.jplceditor.project.io.LDParser;  
import java.io.BufferedWriter;  
import java.io.File;  
import java.io.IOException;  
import java.nio.charset.Charset;  
import java.nio.file.Files;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.logging.Level;  
import java.util.logging.Logger;
```

```
/**
 *
 * @author Vusivus
 */
public class JPLCProject {

    private String path;
    private String projectName;
    private List<LadderDiagram> ladderDiagrams;
    private List<PLCOOperand> tags;
    private File projectFile, hexFile, asmFile, ldFile, programFile;
    private LDCompiler compiler;
    private static final Charset CHARSET = Charset.forName("UTF-8");
    private SessionManager plcSession;
    private boolean compiled;

    /**
     *
     * @param path
     * @param projectName
     */
    public JPLCProject(String path, String projectName) {
        this.path = path + projectName + "\\";
        this.projectName = projectName;
        File dir = new File(this.path);
        dir.mkdir();
        ladderDiagrams = new ArrayList<>();
        tags = new ArrayList<>();
        createDefault();
    }

    /**
     *
     * @param plcProject
     */
    public JPLCProject(SavableProject plcProject) {
        path = plcProject.directory;
        projectName = plcProject.getProjectTitle();
        tags = plcProject.getTagList();
        ladderDiagrams = fromIOLDList(plcProject.getLadderDiagramList());
        projectFile = new File(path + projectName + ".xml");
    }

    private List<IOLD> fromLadderDiagramList() {
        List<IOLD> list = new ArrayList<>();
        ladderDiagrams.stream().forEach((ld) -> {
            list.add(LDParser.fromLadderDiagram(ld));
        });
        return list;
    }

    private List<LadderDiagram> fromIOLDList(List<IOLD> iolds) {
        List<LadderDiagram> list = new ArrayList<>();
        iolds.stream().forEach((ld) -> {
            list.add(LDParser.fromIOLD(ld));
        });
    }
}
```

```

        return list;
    }

private void createDefault() {
    ladderDiagrams.add(new LadderDiagram());
    projectFile = new File(path + projectName + ".xml");
    asmFile = new File(path + projectName + ".asm");
    hexFile = new File(path + projectName + ".hex");
    ldFile = new File(path + projectName + ".ld");
    programFile = new File(path + projectName + ".pgm");
    compiler = new LDCompiler();
}

return ladderDiagrams.get(0);
}

new LadderDiagram());
}

new PLCOperand(type, name));
}

return path;
}



```

```
public List<LadderDiagram> getLadderDiagrams() {
    return ladderDiagrams;
}

/**
 *
 * @return
 */
public List<PLCOperand> getTags() {
    return tags;
}

/**
 *
 * @param path
 */
public void setPath(String path) {
    this.path = path;
}

/**
 *
 * @return
 */
public String getProjectName() {
    return projectName;
}

/**
 *
 * @param projectName
 */
public void setProjectName(String projectName) {
    this.projectName = projectName;
}

/**
 *
 * @return @throws IOException
 */
public StringBuilder saveProject() throws IOException {
    StringBuilder status = new StringBuilder();
    status.append("\n");
    XStream xs = XmlAlias.projectAlias();
    List<IOLD> ldList = fromLadderDiagramList();
    SavableProject p = new SavableProject();
    p.setDirectory(path);
    p.setLadderDiagramList(ldList);
    p.setProjectTitle(projectName);
    p.setTagList(tags);
    String content = xs.toXML(p);
    XmlFile.writeFile(content, projectFile);
    status.append("-----\n");
    status.append("\n");
    status.append("Saved project ").append(projectName).append(" at ");
    status.append(path);
}
```

```

        status.append("\n");
        return status;
    }

    /**
     *
     * @return
     */
    public StringBuilder compileProject() {
        StringBuilder status = new StringBuilder();
        // if (!compiled) {
        if (compiler.compile(ladderDiagrams.get(0), tags)) {
            try {
                status.append("\n");

status.append("=====");
                status.append("\n");
                status.append("Compiled Project ").append(projectName);
                status.append("\n");
                saveFile(compiler.getLDStatement(), ldFile);
                status.append("Generated Ladder Diagram File at
:").append(ldFile.getAbsolutePath());
                status.append("\n");
                saveFile(compiler.getPicAsm(), asmFile);
                status.append("Generated asm File at
:").append(asmFile.getAbsolutePath());
                status.append("\n");
                saveFile(compiler.getPicHex(), hexFile);
                status.append("Generated Hex File at
:").append(hexFile.getAbsolutePath());
                status.append("\n");
                saveFile(compiler.getPicProgramMap().toString(),
programFile);
                status.append("Generated Program File at
:").append(programFile.getAbsolutePath());

            } catch (IOException ex) {

Logger.getLogger(JPLCProject.class.getName()).log(Level.SEVERE, null, ex);
            }
            compiled = true;
        } else {
            status.append("\n");

status.append("=====");
            status.append("\nError compiling program");
            compiled =false;
        }

        // }
        return status;
    }

    /**
     *
     * @param file
     * @return
     */

```

```
/*
public static SavableProject readInProject(File file) {
    XStream xs = XmlAlias.projectAlias();
    SavableProject project = (SavableProject) xs.fromXML(file);
    return project;
}

/**
 *
 * @param content
 * @param file
 * @throws IOException
 */
public static void saveFile(String content, File file) throws IOException
{
    BufferedWriter writer = Files.newBufferedWriter(file.toPath(),
CHARSET);
    writer.write(content, 0, content.length());
    writer.close();
}

/**
 *
 * @return
 */
public StringBuilder download() {
    StringBuilder status = new StringBuilder();
    status.append("\n");
    if (compiled) {
        plcSession = new SessionManager();
        Pic18Prog program = compiler.getPicProgram();
        status.append(program.getProgramInformation());

status.append(plcSession.writeToPlc(program.getCodeHex().array()));
    } else {
        status.append("=====      ERROR !!!!!      ======");
        status.append("\n");
        status.append("Project ").append(this.projectName).append(" is
not compiled yet !!!!");
        status.append("\n");
    }
    return status;
}
```

```
package com.vts.jplceditor.project;

import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.project.io.IOLD;
import java.util.List;


$$/**
 *
 * @author Vusivus$$

```

```
 */
public class SavableProject {

    String directory;
    String projectTitle;
    List<IOLD> ladderDiagramList;
    List<PLCOperand> tagList;

    /**
     *
     * @return
     */
    public String getDirectory() {
        return directory;
    }

    /**
     *
     * @param directory
     */
    public void setDirectory(String directory) {
        this.directory = directory;
    }

    /**
     *
     * @return
     */
    public String getProjectTitle() {
        return projectTitle;
    }

    /**
     *
     * @param projectTitle
     */
    public void setProjectTitle(String projectTitle) {
        this.projectTitle = projectTitle;
    }

    /**
     *
     * @return
     */
    public List<IOLD> getLadderDiagramList() {
        return ladderDiagramList;
    }

    /**
     *
     * @param ladderDiagramList
     */
    public void setLadderDiagramList(List<IOLD> ladderDiagramList) {
        this.ladderDiagramList = ladderDiagramList;
    }

}
```

```
 *
 * @return
 */
public List<PLCOperand> getTagList() {
    return tagList;
}

/**
 *
 * @param tagList
 */
public void setTagList(List<PLCOperand> tagList) {
    this.tagList = tagList;
}

}

package com.vts.jplceditor.project;

import com.thoughtworks.xstream.XStream;
import com.vts.jplceditor.compiler.plc.PLCOperand;
import com.vts.jplceditor.project.io.IOInstruction;
import com.vts.jplceditor.project.io.IOLD;
import com.vts.jplceditor.project.io.IONode;
import com.vts.jplceditor.project.io.IORung;

/**
 *
 * @author Vusivus
 */
public class XmlAlias {

    /**
     *
     * @return
     */
    public static XStream projectAlias() {
        XStream xs = new XStream();
        xs.alias("Project", SavableProject.class);
        xs.alias("LadderDiagram", IOLD.class);
        xs.alias("Rung", IORung.class);
        xs.alias("Node", IONode.class);
        xs.alias("Instruction", IOInstruction.class);
        xs.alias("Parameter", PLCOperand.class);
        xs.alias("Tag", PLCOperand.class);
        return xs;
    }
}

package com.vts.jplceditor.project.
```

```
import com.vts.jplceditor.JPlcEditor;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.IOException;
import java.nio.charset.Charset;
import java.nio.file.Files;
import java.util.prefs.Preferences;

/**
 *
 * @author Electronics
 */
public class XmlFile {

    /**
     * The character set. UTF-8 works good for windows, mac and Umlaute.
     */
    private static final Charset CHARSET = Charset.forName("UTF-8");
    private static final String xmlHeader = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>";
    /**
     * Reads the specified file and returns the content as a String.
     *
     * @param file
     * @return
     * @throws IOException thrown if an I/O error occurs opening the file
     */
    public static String readFile(File file) throws IOException {
        StringBuffer stringBuffer = new StringBuffer();

        BufferedReader reader = Files.newBufferedReader(file.toPath(), CHARSET);

        String line = null;
        while ((line = reader.readLine()) != null) {
            stringBuffer.append(line);
        }

        reader.close();

        return stringBuffer.toString();
    }

    /**
     * Saves the content String to the specified file.
     *
     * @param content
     * @param file
     * @throws IOException thrown if an I/O error occurs opening or creating
     * the file
     */
    public static void saveFile(String content, File file) throws IOException {
        BufferedWriter writer = Files.newBufferedWriter(file.toPath(), CHARSET);
        writer.write(xmlHeader);
        writer.write(content);
        writer.close();
    }
}
```

```
        writer.write("\n");
        writer.write(content, 0, content.length());
        writer.close();
    }
}

/**
 * Returns the person file preference, i.e. the file that was last opened.
 * The preference is read from the OS specific registry. If no such
 * preference can be found, null is returned.
 *
 * @return
 */
public File getPersonFilePath() {
    Preferences prefs = Preferences.userNodeForPackage(JPlcEditor.class);
    String filePath = prefs.get("filePath", null);
    if (filePath != null) {
        return new File(filePath);
    } else {
        return null;
    }
}

/**
 * Sets the file path of the currently loaded file.
 * The path is persisted in the OS specific registry.
 *
 * @param file the file or null to remove the path
 */
public void setPersonFilePath(File file) {
    Preferences prefs = Preferences.userNodeForPackage(JPlcEditor.class);
    if (file != null) {
        prefs.put("filePath", file.getPath());
    }
}
```

Appendix H: JPLC Editor Communication layer classes code

```
package com.vts.jplceditor.comms.usb;

import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.Callable;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;
import java.util.logging.Level;
import java.util.logging.Logger;
import purejavahidapi.DeviceRemovalListener;
import purejavahidapi.HidDevice;
import purejavahidapi.HidDeviceInfo;
import purejavahidapi.InputReportListener;
import purejavahidapi.PureJavaHidApi;

/**
 *
 * @author Vusman
 */
public class HidManager implements AutoCloseable {

    private final int vendorId = 0x1234;
    private final int productId = 0x001;
    private final byte inputBuffer = 64;
    private final byte outputBuffer = 64;
    private final String vendorName = "VT_Labs";
    private final String productName = "VT_PLC";
    private HidDeviceInfo devInfo;
    private HidDevice device;
    private boolean deviceFound, connected, ack;
    private ByteBuffer receiveBuf;
    private final ByteBuffer transmitBuf;
    private short receiveLen;
    private List<ByteBuffer> readData;
    private ExecutorService executor;

    /**
     *
     */
    public HidManager() {
        transmitBuf = ByteBuffer.allocate(64);
        receiveBuf = ByteBuffer.allocate(128);
        transmitBuf.order(ByteOrder.BIG_ENDIAN);
        receiveBuf.order(ByteOrder.BIG_ENDIAN);
        executor = Executors.newSingleThreadExecutor();
    }
}
```

```

}

/**
 *
 * @param feedback
 * @return
 */
public boolean locatePlc(StringBuilder feedback) {
    List<HidDeviceInfo> devList = PureJavaHidApi.enumerateDevices();
    for (HidDeviceInfo info : devList) {
        if (info.getVendorId() == (short) vendorId && info.getProductId()
== (short) productId) {
            devInfo = info;
            deviceFound = true;
            feedback.append("\nPLC Device found");
            return true;
        }
    }
    feedback.append("\nCould Not find PLC device");
    deviceFound = false;
    return false;
}

/**
 *
 * @param feedback
 * @return
 */
public boolean connectPlc(StringBuilder feedback) {
    if (connected) {
        shutdownAndAwaitTermination(executor);
        executor = Executors.newSingleThreadExecutor();
        return true;
    }
    feedback.append("\n");
    if (deviceFound) {
        try {
            device = PureJavaHidApi.openDevice(devInfo);
            device.setInputReportListener(reportListener);
            device.setDeviceRemovalListener(removalListener);
            //Create Packet to connect to PLC
            PlcMessage connect = new PlcMessage(PlcMessage.cmdCONNECT);
            transmitBuf.clear(); //clear transmit buffer
            PlcFormatter.formatMessage(connect, transmitBuf);
            connected = write64ToPlc(transmitBuf.array());
            feedback.append("\nSent output report requesting connection
to ");
        }

        feedback.append(device.getHidDeviceInfo().getProductString());
        feedback.append("\nWaiting for response...");
        byte code = 0;
        if (connected) {
            feedback.append("\n Received Response from PLC");
            code = receiveBuf.get(2);
            feedback.append("\nResponse code:
").append(String.format("0x%02X", code));
        } else {
    }
}

```

```

        feedback.append("\nDevice Timed Out. Could not get
response from PLC");
    }
    connected = code == 0x06;
} catch (IOException ex) {

Logger.getLogger(HidManager.class.getName()).log(Level.SEVERE, null, ex);
}
} else {
    feedback.append("Device Not Found Yet. You need to locate PLC
first");
}
return connected;
}

/**
 *
 * @param data
 * @return
 */
public boolean write64ToPlc(byte[] data) {
    boolean responded = false;
    if (deviceFound) {
        try {
            receiveLen = 0;
            ack = false;
            Future<Boolean> responseResult = executor.submit(new
UsbPortResponse());
            device.setOutputReport((byte) 0, data, 64);
            responded = responseResult.get(2000, TimeUnit.MILLISECONDS);
        } catch (InterruptedException | ExecutionException |
TimeoutException ex) {
            //shutdownAndAwaitTermination(executor);

Logger.getLogger(HidManager.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    return responded;
}

/**
 *
 * @param message
 * @return
 * @throws Exception
 */
public boolean sendCmd(PlcMessage message) throws Exception {
    boolean responded = false;
    if (connected) {
        for (int i = 0; i < 64; i++) {
            transmitBuf.put(i, (byte) 0);
        }
        transmitBuf.clear();
        PlcFormatter.formatMessage(message, transmitBuf);
        responded = write64ToPlc(transmitBuf.array());
    }
    return responded;
}

```

```

    }

    /**
     *
     * @param messages
     * @param feedback
     * @return
     */
    public boolean handleBulkTransfer(List<PlcMessage> messages,
StringBuilder feedback) {
    boolean responded;
    if (connected) {
        int size = messages.size();
        feedback.append("\nWriting To PLC.");
        for (int k = 0; k < size; k++) {
            try {
                //first empty the transmit buff
                for (int i = 0; i < 64; i++) {
                    transmitBuf.put(i, (byte) 0);
                }
                transmitBuf.clear();
                //Encode each message
                PlcFormatter.formatMessage(messages.get(k), transmitBuf);
                feedback.append("\nSent packet number
:").append(k).append(". Waiting for response");
                receiveLen = 0;
                ack = false;
                //send the packet
                Future<Boolean> responseResult = executor.submit(new
UsbPortResponse());
                device.setOutputReport((byte) 0, transmitBuf.array(),
64);
                responded = responseResult.get(5000,
TimeUnit.MILLISECONDS);
                if (!responded) {
                    feedback.append("\nPLC did not acknowledge request.
Ending session");
                    return false;
                }
                ack = receiveBuf.get(2) == 0x06;
                if (ack) {
                    feedback.append("\nPLC has acknowledged request.");
                    Thread.sleep(500);
                } else if (!ack) {
                    feedback.append("\nPLC has responded but did not
acknowledge request.");
                    return false;
                }
            } catch (InterruptedException | ExecutionException |
TimeoutException ex) {
                shutdownAndAwaitTermination(executor);
                Logger.getLogger(HidManager.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
    return ack;
}

```

```

}

/**
 *
 * @param messages
 * @param feedback
 * @return
 */
public boolean handleBulkRead(List<PlcMessage> messages, StringBuilder
feedback) {
    boolean responded = false;
    readData = new ArrayList<>();
    int size = messages.size();
    feedback.append("\nReading From PLC.");
    for (int k = 0; k < size; k++) {
        try {
            for (int i = 0; i < 64; i++) {
                transmitBuf.put(i, (byte) 0);
            }
            transmitBuf.clear();
            PlcFormatter.formatMessage(messages.get(k), transmitBuf);
            feedback.append("\nSent packet number :").append(k).append(".");
Waiting for response");
            receiveLen = 0;
            Future<Boolean> responseResult = executor.submit(new
UsbPortResponse());
            device.setOutputReport((byte) 0, transmitBuf.array(), 64);
            responded = responseResult.get(5000, TimeUnit.MILLISECONDS);
            if (!responded) {
                feedback.append("\nPLC did not acknowledge request. Ending
session");
                break;
            } else {
                feedback.append("\nPLC has acknowledged request.");
                readData.add(receiveBuf.duplicate());
                Thread.sleep(500);
            }
        } catch (InterruptedException | ExecutionException |
TimeoutException ex) {
            shutdownAndAwaitTermination(executor);

Logger.getLogger(HidManager.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    return responded;
}

private final DeviceRemovalListener removalListener = new
DeviceRemovalListener() {
    @Override
    public void onDeviceRemoval(HidDevice hd) {
        System.out.println("Device Removed");
        if (executor != null) {
            shutdownAndAwaitTermination(executor);
        }
    }
};
}

```

```

    private final InputReportListener reportListener = new
InputReportListener() {
    @Override
    public void onInputReport(HidDevice hd, byte id, byte[] bytes, int
len) {
        if (len == 64) {
            receiveBuf.clear();
            System.out.print("\nReceived " + len + " bytes with ID " + id
+ ": ");
            for (int i = 0; i < len; i++) {
                receiveBuf.put(i, bytes[i]);
                System.out.printf("%02X ", bytes[i]);
            }
        }
        receiveLen = (short) len;
    }
};

/**
 *
 */
private void usbShutdown() {
    if (executor != null) {
        shutdownAndAwaitTermination(executor);
    }
    if (connected) {
        device.close();
    }
}

private void shutdownAndAwaitTermination(ExecutorService pool) {
    pool.shutdown(); // Disable new tasks from being submitted
    try {
        // Wait a while for existing tasks to terminate
        if (!pool.awaitTermination(60, TimeUnit.SECONDS)) {
            pool.shutdownNow(); // Cancel currently executing tasks
            // Wait a while for tasks to respond to being cancelled
            if (!pool.awaitTermination(60, TimeUnit.SECONDS)) {
                System.err.println("Pool did not terminate");
            }
        }
    } catch (InterruptedException ie) {
        // (Re-)Cancel if current thread also interrupted
        pool.shutdownNow();
        // Preserve interrupt status
        Thread.currentThread().interrupt();
    }
}

/**
 *
 * @return
 */
public boolean isConnected() {
    return connected;
}

```

```
/*
 *
 * @return
 */
public boolean isAck() {
    return ack;
}

/**
 *
 * @return
 */
public short getReceiveLen() {
    return receiveLen;
}

/**
 *
 * @return
 */
public ByteBuffer getTransmitBuf() {
    return transmitBuf;
}

/**
 *
 * @return
 */
public List<ByteBuffer> getReadData() {
    return readData;
}

@Override
public void close() throws Exception {
    System.out.println("Device Closed");
    usbShutdown();
}

class UsbPortResponse implements Callable<Boolean> {

    @Override
    public Boolean call() throws Exception {
        receiveLen = 0;
        while (receiveLen == 0) {
            Thread.sleep(100);
        }
        return receiveLen > 0;
    }
}

package com.vts.jplceditor.comms.usb;
```

```
import java.nio.ByteBuffer;

/**
 *
 * @author Electronix
 */
public abstract class PlcFormatter {

    //DF1 Protocol bytes
    static final short DLE = 0x10;
    static final short STX = 0x02;
    private static final short ENQ = 0x05;
    private static final short ACK = 0x06;
    private static final short NAK = 0x0F;
    static final short ETX = 0x03;

    /**
     *
     * @param msg
     * @param transmitBuf
     * @return
     */
    public static int formatMessage(PlcMessage msg, ByteBuffer transmitBuf) {
        int length = 0;
        short cmd = msg.getCmd();
        transmitBuf.rewind();
        transmitBuf.put((byte) DLE);
        transmitBuf.put((byte) STX);
        switch (cmd) {
            case PlcMessage.cmdCONNECT:
                transmitBuf.put((byte) cmd);
                transmitBuf.put((byte) DLE);
                transmitBuf.put((byte) ETX);
                length = 5;
                break;
            case PlcMessage.cmdERASE:
                transmitBuf.put((byte) cmd);
                transmitBuf.putShort((short) msg.getAddress());
                transmitBuf.put((byte) msg.getLength());
                transmitBuf.put((byte) DLE);
                transmitBuf.put((byte) ETX);
                length = 8;
                break;
            case PlcMessage.cmdWRITE_FLASH:
                transmitBuf.put((byte) cmd);
                transmitBuf.putShort((short) msg.getAddress());
                transmitBuf.put((byte) msg.getLength());
                transmitBuf.put(msg.getData());
                transmitBuf.put((byte) DLE);
                transmitBuf.put((byte) ETX);

                length = 40;
                break;
            case PlcMessage.cmdREAD_FLASH:
                transmitBuf.put((byte) cmd);
                transmitBuf.putShort((short) msg.getAddress());
        }
    }
}
```

```
        transmitBuf.put((byte) msg.getLength());
        transmitBuf.put((byte) DLE);
        transmitBuf.put((byte) ETX);

        length = 8;
        break;

    }
    return length;
}
}

package com.vts.jplceditor.comms.usb;

/**
 *
 * @author Electronix
 */
public class PlcMessage {

    private final short cmd;
    private int address;
    private short length;
    private byte[] data;

    /**
     *
     */
    public final static short cmdCONNECT = 0x06;

    /**
     *
     */
    public final static short cmdREAD_FLASH = 0x02;

    /**
     *
     */
    public final static short cmdREAD_RAM = 0x03;

    /**
     *
     */
    public final static short cmdWRITE_FLASH = 0x01;

    /**
     *
     */
    public final static short cmdWRITE_RAM = 0x04;

    /**
     *
     */
    public final static short cmdERASE = 0x05;
```

```
/**  
 *  
 * @param cmd  
 */  
public PlcMessage(short cmd) {  
    this.cmd = cmd;  
}  
  
/**  
 *  
 * @param address  
 */  
public void setAddress(int address) {  
    this.address = address;  
}  
  
/**  
 *  
 * @param length  
 */  
public void setLength(short length) {  
    this.length = length;  
}  
  
/**  
 *  
 * @param data  
 */  
public void setData(byte[] data) {  
    this.data = data;  
}  
  
/**  
 *  
 * @return  
 */  
public short getCmd() {  
    return cmd;  
}  
  
/**  
 *  
 * @return  
 */  
public long getAddress() {  
    return address;  
}  
  
/**  
 *  
 * @return  
 */  
public short getLength() {  
    return length;  
}
```

```

    /**
     *
     * @return
     */
    public byte[] getData() {
        return data;
    }

}

package com.vts.jplceditor.comms.usb;

import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import java.nio.ByteBuffer;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusman
 */
public class SessionManager {

    private int transactionHandle;
    private short status;
    private short sessionCmd;
    private byte sessionNumber;
    private HidManager usb;
    private int replySize;
    private boolean readingRam;

    /**
     *
     */
    public SessionManager() {
        init();
    }

    private void init() {
        usb = new HidManager();
    }

    /**
     *
     * @param packet
     * @return
     */
    public StringBuilder writeToPlc(byte[] packet) {
        StringBuilder feedback = new StringBuilder();

        if (usb.locatePlc(feedback)) {//See if PLC Device is connected.
            if (usb.connectPlc(feedback)) {//Request connection from device
                feedback.append("Writing to PLC.");
                List<PlcMessage> messages = new ArrayList<>();

```

```

        createFlashWritePackets(packet, messages);
        if (usb.handleBulkTransfer(messages, feedback)) {
            List<PlcMessage> readMessages = new ArrayList<>();
            createFlashReadPackets(packet, readMessages);
            usb.handleBulkRead(readMessages, feedback);
            verifyProgram(usb.getReadData(), packet, feedback);
        }
    }
} else {
    feedback.append("Could not locate PLC");
}
return feedback;
}

public StringBuilder readFromPlcRam(ByteBuffer io, ByteBuffer bank1,
ByteBuffer bank2) {
    StringBuilder feedback = new StringBuilder();

    if (usb.locatePlc(feedback)) {//See if PLC Device is connected.
        if (usb.connectPlc(feedback)) {//Request connection from device
            feedback.append("\nReading from PLC RAM.");
            List<PlcMessage> messages = createRamReadPackets(io, bank1,
bank2);
            readingRam = true;
            while (readingRam) {
                if(usb.handleBulkRead(messages, feedback)){
                    List<ByteBuffer> readData = usb.getReadData();
                    io.put(readData.get(0));
                    // bank1.put(readData.get(1), replySize, replySize);
                }
            }
        }
    } else {
        feedback.append("Could not locate PLC");
    }
    return feedback;
}

/**
 *
 * @return
 */
public short getStatus() {
    return status;
}

/**
 *
 * @param packet
 * @param messages
 */
public void createFlashReadPackets(byte[] packet, List<PlcMessage>
messages) {
    int size = packet.length;      //size of the program
    int left = size;              //bytes left to transmit
    int sessions = size / 64;     // number of 64 bit sessions required
}

```

```

        //if times is not multiple of 64 add an extra session for the
fraction
    if (size % 64 > 0) {
        sessions += 1;
    }
    int ptr = 0;
    int address = (int) Pic18Mem.PROGRAM_ORG;
    for (int i = 0; i < sessions; i++) {
        PlcMessage msg = new PlcMessage(PlcMessage.cmdREAD_FLASH);
        msg.setAddress(address);
        msg.setLength((short) 64);
        messages.add(msg);
        address += 64;
    }
}

/**
 *
 * @param packet
 * @param messages
 */
public void createFlashWritePackets(byte[] packet, List<PlcMessage>
messages) {
    int size = packet.length;      //size of the program
    int left = size;              //bytes left to transmit
    int sessions = size / 32;     // number of 64 bit sessions required
    //if times is not multiple of 64 add an extra session for the
fraction
    if (size % 32 > 0) {
        sessions += 1;
    }
    if (size < 32) {
        sessions = 1;
    }

    int ptr = 0;
    int address = (int) Pic18Mem.PROGRAM_ORG;
    PlcMessage erase = new PlcMessage(PlcMessage.cmdERASE);
    erase.setAddress(address);
    erase.setLength((short) sessions);
    messages.add(erase);
    for (int i = 0; i < sessions; i++) {
        PlcMessage msg = new PlcMessage(PlcMessage.cmdWRITE_FLASH);
        byte[] transmitData = new byte[32];
        msg.setAddress(address);
        msg.setLength((short) 32);
        if (left >= 32) {
            for (int k = 0; k < 32; k++) {
                transmitData[k] = packet[ptr];
                ++ptr;
            }
            left = left - 32;
        } else {
            for (int k = 0; k < left; k++) {
                transmitData[k] = packet[ptr];
                ++ptr;
            }
        }
    }
}

```

```

        for (int j = left - 1; j < 32; j++) {
            transmitData[j] = 0;
        }
        left = 0;
    }
    msg.setData(transmitData);
    messages.add(msg);
    address += 32;
}
}

public boolean isReadingRam() {
    return readingRam;
}

public void setReadingRam(boolean readingRam) {
    this.readingRam = readingRam;
}

for (ByteBuffer buffer : readData) {
        if (left >= 64) {
            for (int i = 0; i < 64; i++) {
                if (buffer.get(i) != packet[pointer]) {
                    feedback.append("\nByte number
:").append(pointer).append(" is not the expected byte. Byte is
:").append(String.format("0x%02X", buffer.get(i)));
                }
                ++pointer;
            }
            left = left - 64;
        } else if (left > 0 && left < 64) {
            for (int i = 0; i < left - 1; i++) {
                if (buffer.get(i) != packet[pointer]) {
                    feedback.append("\nByte number
:").append(pointer).append(" is not the expected byte. Byte is
:").append(String.format("0x%02X", buffer.get(i)));
                }
                ++pointer;
            }
            left = 0;
        }
    }
    feedback.append("\nFinished verifying " + size + " bytes.");
}
}

```

```

private List<PlcMessage> createRamReadPackets(ByteBuffer io, ByteBuffer
bank1, ByteBuffer bank2) {
    List<PlcMessage> messages = new ArrayList<>();
    PlcMessage ioRead = new PlcMessage(PlcMessage.cmdREAD_RAM);
    ioRead.setAddress(Pic18Mem.IO_START);
    ioRead.setLength((short) 20);
    messages.add(ioRead);
    if (bank1 != null) {
        int size = bank1.capacity();
        int sessions = size / 56;
        if (size % 56 > 0) {
            sessions += 1;
        }
        if (size < 56) {
            sessions = 1;
        }
        int left = size;
        int addr1 = Pic18Mem.BANK1;
        for (int k = 0; k < sessions; k++) {
            PlcMessage msg = new PlcMessage(PlcMessage.cmdWRITE_RAM);
            msg.setAddress(addr1);
            if (left > 64) {
                msg.setLength((short) 64);
                left -= 64;
            } else {
                msg.setLength((short) left);
                left = 0;
            }
            addr1 += 64;
            messages.add(msg);
        }
    }
    if (bank2 != null) {
        int size = bank2.capacity();
        int sessions = size / 64;
        if (size % 64 > 0) {
            sessions += 1;
        }
        if (size < 64) {
            sessions = 1;
        }
        int left = size;
        int addr1 = Pic18Mem.BANK2;
        for (int k = 0; k < sessions; k++) {
            PlcMessage msg = new PlcMessage(PlcMessage.cmdWRITE_RAM);
            msg.setAddress(addr1);
            if (left > 64) {
                msg.setLength((short) 64);
                left -= 64;
            } else {
                msg.setLength((short) left);
                left = 0;
            }
            addr1 += 64;
            messages.add(msg);
        }
    }
}

```

```
    return messages;
}
}
```

Appendix I: JPLC Editor Compiler layer classes code

```
package com.vts.jplceditor.compiler.ld;

import com.vts.jplceditor.compiler.pic18.Pic18Compiler;
import com.vts.jplceditor.compiler.pic18.Pic18Instruction;
import com.vts.jplceditor.compiler.pic18.Pic18Prog;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.ld.LadderDiagram;
import java.util.List;

/**
 *
 * @author Vusivus
 */
public class LDCompiler {

    private LadderDiagram ld;
    private final LDVisitor visitor;
    private List<PLCOOperand> tags;
    private final Pic18Compiler picCompiler;

    public LDCompiler() {
        visitor = new LDVisitor();
        picCompiler = Pic18Compiler.forPic18f2550();
    }

    public boolean compile(LadderDiagram ladderDiagram, List<PLCOOperand>
tagList) {
        boolean success;
        ld = ladderDiagram;
        tags = tagList;
        if(!visitor.visitLD(ld)){
            success=false;
        }else{
            picCompiler.compileInstructionList(visitor.getRungList(), tagList);
            success=true;
        }
        return success;
    }

    public String getLDStatement(){
        return visitor.generateResult();
    }

    public String getPicAsm(){
        return picCompiler.getAsmOutput().toString();
    }

    public StringBuffer getPicProgramMap(){
        return picCompiler.getPicProgram().getProgramMap();
    }

    public String getPicHex(){
        return picCompiler.getHexOutput().toString();
    }
}
```

```
public Pic18Prog getPicProgram() {
    return picCompiler.getPicProgram();
}
}

package com.vts.jplceditor.compiler.ld;

import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.ld.component.LDRung;
import com.vts.jplceditor.ld.component.LDInstruction;
import com.vts.jplceditor.ld.instruction.Instruction;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusivus
 */
public class LDRungVisitor {

    private final List<LDInstruction> instructionList;
    private LDStatement statement;

    public LDRungVisitor() {
        instructionList = new ArrayList<>();
    }

    public boolean visitRung(LDRung rung) {
        boolean success = false;
        List<LDInstruction> list = rung.getInstructionList();
        for (int i = 0; i < list.size(); i++) {
            if (list.get(i).hasInstruction()) {
                Instruction in = list.get(i).getInstruction();
                if (verifyParameters(in.getPLCInstruction().getOperands() ==
1)) {
                    instructionList.add(list.get(i));
                    success=true;
                }else{
                    list.get(i).select();
                    success=false;
                    break;
                }
            }
        }
        return success;
    }

    public void generateStatement() {
        statement = new LDStatement();
        statement.generateStatement(instructionList);
    }

    public LDStatement getStatement() {
        return statement;
    }
}
```

```

    }

    public List<LDInstruction> getInstructionList() {
        return instructionList;
    }

    private int verifyParameters(List<PLCOperand> operands) {
        int index=-1;
        for(int i=0;i<operands.size();i++){
            if(blank(operands.get(i))){
                index=i;
                break;
            }
        }
        return index;
    }

    private boolean blank(PLCOperand op) {
        return "".equals(op.getIdentity()) || op.getIdentity().equals("Tag_Name");
    }
}

package com.vts.jplceditor.compiler.ld;

import com.vts.jplceditor.compiler.plc.PLCOperand;
import com.vts.jplceditor.ld.component.LDRung;
import com.vts.jplceditor.ld.component.LDInstruction;
import com.vts.jplceditor.ld.instruction.Instruction;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusivus
 */
public class LDRungVisitor {

    private final List<LDInstruction> instructionList;
    private LDStatement statement;

    public LDRungVisitor() {
        instructionList = new ArrayList<>();
    }

    public boolean visitRung(LDRung rung) {
        boolean success = false;
        List<LDInstruction> list = rung.getInstructionList();
        for (int i = 0; i < list.size(); i++) {
            if (list.get(i).hasInstruction()) {
                Instruction in = list.get(i).getInstruction();
                if(verifyParameters(in.getPLCInstruction().getOperands())==1){
                    instructionList.add(list.get(i));
                }
            }
        }
        return success;
    }
}

```

```

        success=true;
    }else{
        list.get(i).select();
        success=false;
        break;
    }
}
return success;
}

public void generateStatement() {
    statement = new LDStatement();
    statement.generateStatement(instructionList);
}

public LDStatement getStatement() {
    return statement;
}

public List<LDInstruction> getInstructionList() {
    return instructionList;
}

private int verifyParameters(List<PLCOperand> operands) {
    int index=-1;
    for(int i=0;i<operands.size();i++){
        if(blank(operands.get(i))){
            index=i;
            break;
        }
    }
    return index;
}

private boolean blank(PLCOperand op) {
    return "".equals(op.getIdentity()) || op.getIdentity().equals("Tag_Name");
}

}

package com.vts.jplceditor.compiler.ld;

import com.vts.jplceditor.compiler.plc.PLCOperand;
import com.vts.jplceditor.ld.component.LDRung;
import com.vts.jplceditor.ld.component.LDInstruction;
import com.vts.jplceditor.ld.instruction.Instruction;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusivus
 */

```

```

public class LDRungVisitor {

    private final List<LDInstruction> instructionList;
    private LDStatement statement;

    public LDRungVisitor() {
        instructionList = new ArrayList<>();
    }

    public boolean visitRung(LDRung rung) {
        boolean success = false;
        List<LDInstruction> list = rung.getInstructionList();
        for (int i = 0; i < list.size(); i++) {
            if (list.get(i).hasInstruction()) {
                Instruction in = list.get(i).getInstruction();
                if (verifyParameters(in.getPLCInstruction().getOperands()) == -1) {
                    instructionList.add(list.get(i));
                    success=true;
                }else{
                    list.get(i).select();
                    success=false;
                    break;
                }
            }
        }
        return success;
    }

    public void generateStatement() {
        statement = new LDStatement();
        statement.generateStatement(instructionList);
    }

    public LDStatement getStatement() {
        return statement;
    }

    public List<LDInstruction> getInstructionList() {
        return instructionList;
    }

    private int verifyParameters(List<PLCOOperand> operands) {
        int index=-1;
        for(int i=0;i<operands.size();i++){
            if(blank(operands.get(i))){
                index=i;
                break;
            }
        }
        return index;
    }

    private boolean blank(PLCOOperand op) {
        return "".equals(op.getIdentity()) || op.getIdentity() == "Tag_Name";
    }
}

```

}

```
package com.vts.jplceditor.compiler.ld;

import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.ld.component.LDRung;
import com.vts.jplceditor.ld.component.LDInstruction;
import com.vts.jplceditor.ld.instruction.Instruction;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusivus
 */
public class LDRungVisitor {

    private final List<LDInstruction> instructionList;
    private LDStatement statement;

    public LDRungVisitor() {
        instructionList = new ArrayList<>();
    }

    public boolean visitRung(LDRung rung) {
        boolean success = false;
        List<LDInstruction> list = rung.getInstructionList();
        for (int i = 0; i < list.size(); i++) {
            if (list.get(i).hasInstruction()) {
                Instruction in = list.get(i).getInstruction();
                if (verifyParameters(in.getPLCInstruction().getOperands() ==
1)) {
                    instructionList.add(list.get(i));
                    success=true;
                }else{
                    list.get(i).select();
                    success=false;
                    break;
                }
            }
        }
        return success;
    }

    public void generateStatement() {
        statement = new LDStatement();
        statement.generateStatement(instructionList);
    }

    public LDStatement getStatement() {
        return statement;
    }

    public List<LDInstruction> getInstructionList() {
```

```

        return instructionList;
    }

    private int verifyParameters(List<PLCOperand> operands) {
        int index=-1;
        for(int i=0;i<operands.size();i++){
            if(blank(operands.get(i))){
                index=i;
                break;
            }
        }
        return index;
    }

    private boolean blank(PLCOperand op) {
        return "".equals(op.getIdentity()) || op.getIdentity().equals("Tag_Name");
    }

}

package com.vts.jplceditor.compiler.pic18;

/**
 * Represents a PIC18 Byte Oriented Instruction
 *
 * @author Vusivus
 */
public class ByteInstruction implements Pic18Instruction {

    private final boolean d;
    private final boolean a;
    private short file, file2;
    private final Opcode op;
    private short bitCode, bitCode2;
    private int wordCount;

    /**
     * Instantiates a Byte Oriented Instruction
     *
     * @param opcode
     * <p>
     * The Byte oriented Opcode</p>
     *
     * @param file 8 bit file register address
     * @param a
     * <p>
     * a=0(false) to force Access Bank</p>
     * <p>
     * a=1(true) for BSR or select Bank</p>
     * <p>
     * If the opcode doesn't use a set it to false</p>
     * @param d<p>
     * d=0(false) for result destination to be WREG</p>
     * <p>
     * d=1(true) for result destination to be file</p>
     * <p>

```

```

        * If the opcode doesn't use d set it to false</p>
        */
    public ByteInstruction(Opcode op, int file, boolean a, boolean d) {
        this.d = d;
        this.a = a;
        this.file = (short) file;
        this.op = op;
    }

    /**
     * Assembles the instruction into a 16 bit code
     */
    @Override
    public void Assemble() {
        bitCode = op.getHexCode();

        if (op.getBits() == 6 || op.getBits() == 7) {
            if (d == true) {
                bitCode |= 0x02;
            }
            if (a == true) {
                bitCode |= 0x01;
            }
            bitCode <<= 8;
            bitCode |= (file & 0xFF);
            wordCount = 1;
        } else if (op.getBits() == 8) {
            bitCode = (short) (op.getHexCode() & 0xF0);
            bitCode <<= 8;
            bitCode |= (file & 0xFFFF);
            bitCode2 = (short) 0xF000;
            bitCode2 |= (file2 & 0xFFFF);
            wordCount = 2;
        }
    }

    /**
     * change the file address
     *
     * @param file the new 8 bit address
     */
    public void setFile(int file) {
        this.file = (short) file;
    }

    /**
     * Set the file address for two word Byte Oriented Instructions like
     MOVFF
     *
     * @param file2 8 bit address of destination file
     */
    @Override
    public void setFile2(int file2) {
        this.file2 = (short) file2;
    }

    /**

```

```

/*
 * @return the assembled 16 bit PIC code for the Instruction
 */
@Override
public short get16BitCode() {
    return bitCode;
}

/**
 *
 * @return the second word of the assembled 16 bit PIC code for the
 * Instruction
 */
@Override
public short getSecond16BitCode() {
    return bitCode2;
}

/**
 *
 * @return the number of 16 bit words in the instruction
 */
@Override
public int getWordCount() {
    return wordCount;
}

@Override
public String toString() {
    if (wordCount == 1) {
        return instrStr();
    } else {
        return instrStr();
    }
}

private String instrStr() {
    int d = 0, a = 0;
    if (this.d == true) {
        d = 1;
    }
    if (this.a == true) {
        a = 1;
    }
    if (op.getBits() == 6) {
        return op.name() + " " + String.format("0x%02X", (file)) + ", " +
d + ", " + a;
    }
    if (op.getBits() == 7) {
        return op.name() + " " + String.format("0x%02X", (file)) + ", " +
a;
    } else {
        return op.name() + " " + String.format("0x%02X", (file)) + ", " +
String.format("0x%02X", (file2));
    }
}
}

```

```
package com.vts.jplceditor.compiler.pic18;

/**
 *
 * @author Vusivus
 */
public class ControlInstruction implements Pic18Instruction {

    private Opcode opcode;
    private boolean s;
    private int n;
    private short bitCode, bitCode2;
    private int wordCount;

    public ControlInstruction(Opcode opcode) {
        this.opcode = opcode;
    }

    public ControlInstruction(Opcode opcode, int n) {
        this.opcode = opcode;
        this.n = n;
    }

    public ControlInstruction(Opcode opcode, boolean s) {
        this.opcode = opcode;
        this.s = s;
    }

    public ControlInstruction(Opcode opcode, boolean s, int n) {
        this.opcode = opcode;
        this.s = s;
        this.n = n;
    }

    @Override
    public void Assemble() {
        if (null != opcode) switch (opcode) {
            case GOTO:{
                bitCode = (short) 0xEF00;
                bitCode |= (byte) n & 0xFF;
                int temp = (short) (n & 0xFFFF00);
                bitCode2 = (short) (temp >> 8);
                bitCode2 |= 0xF0000;
                wordCount = 2;
                break;
            }
            case BRA:{  
                bitCode = (short) 0xD000;  
                bitCode |= (short) n & 0x7FF;  
                wordCount = 1;  
                break;  
            }
            case CALL:{  
        }
    }
}
```

```
        bitCode = (short) 0xEC00;
        if (s) {
            bitCode |= 0x0100;
        } else
            bitCode |= (byte) n & 0xFF;
        int temp = (short) (n & 0xFFFF00);
        bitCode2 = (short) (temp >> 8);
        bitCode2 |= 0xF0000;
        wordCount = 2;
        break;
    }
case NOP:
    bitCode = 0;
    wordCount = 1;
    break;
case RETURN:
    bitCode = 0x12;
    if (s) {
        bitCode |= 0x1;
    } else
        wordCount=1;
    break;
default:
    break;
}
}

@Override
public short get16BitCode() {
    return bitCode;
}

@Override
public short getSecond16BitCode() {
    return bitCode2;
}

@Override
public int getWordCount() {
    return wordCount;
}

@Override
public void setFile(int file) {
    n = file;
}

@Override
public void setFile2(int file) {

}

@Override
public String toString() {
    if (wordCount == 1) {
        return instrStr();
    } else {
        return instrStr();
    }
}
```

```
}

private String instrStr() {
    int s1 = 0;
    if (s) {
        s1 = 1;
    }
    switch (opcode.getBits()) {
        case 8:
            return opcode.name() + " " + String.format("0x%02X", (n));
        case 12:
            return opcode.name() + " " + String.format("0x%05X", (n));
        case 11:
            return opcode.name() + " " + String.format("0x%05X", (n)) + "
" + s1;
        case 5:
            return opcode.name() + " " + String.format("0x%03X", (n));
        case 15:
            return opcode.name() + " " + s1;
        default:
            return opcode.name() + " ";
    }
}

public byte reverseByte(byte x) {
    int intSize = 8;
    byte y = 0;
    for (int position = intSize - 1; position > 0; position--) {
        y += ((x & 1) << position);
        x >>= 1;
    }
    return y;
}
```

```
package com.vts.jplceditor.compiler.pic18;

/**
 *
 * @author Vusivus
 */
public class LiteralInstruction implements Pic18Instruction {

    private final Opcode opcode;
    private short literal;
    private short bitCode, bitCode2;
    private int wordCount;
    private byte f;

    public LiteralInstruction(Opcode opcode, int literal) {
        this.opcode = opcode;
        this.literal = (short) literal;
    }
}
```

```

    }

    public LiteralInstruction(Opcode opcode, byte f, int literal) {
        this.opcode = opcode;
        this.literal = (short) (literal & 0xFFFF);
        this.f = (byte) (f & 0x03);
    }

    @Override
    public void Assemble() {
        bitCode = opcode.getHexCode();
        switch (opcode.getBits()) {
            case 8:
                bitCode <= 8;
                bitCode += literal & 0xFF;
                wordCount = 1;
                break;
            case 12:
                bitCode <= 4;
                bitCode |= literal & 0xF;
                wordCount = 1;
                break;
            case 10:
                bitCode |= f;
                bitCode <= 4;
                bitCode |= (literal & 0xF00) >> 8;
                bitCode2 = (short) 0xF000;
                bitCode2 |= literal & 0xFF;
                wordCount = 2;
                break;
            default:
                break;
        }
    }

    @Override
    public short get16BitCode() {
        return bitCode;
    }

    @Override
    public short getSecond16BitCode() {
        return bitCode2;
    }

    @Override
    public int getWordCount() {
        return wordCount;
    }

    @Override
    public void setFile2(int file) {
        literal = (short) file;
    }

    @Override
    public void setFile(int file) {

```

```

    }

    public void setF(byte f) {
        this.f = (byte) (f * 0x03);
    }

    @Override
    public String toString() {
        if (wordCount == 1) {
            return instrStr();
        } else {
            return instrStr();
        }
    }

    private String instrStr() {
        if (opcode.getBits() == 8) {
            return opcode.name() + " " + String.format("0x%02X", (literal));
        }
        if (opcode.getBits() == 12) {
            return opcode.name() + " " + String.format("0x%01X", (literal));
        } else {
            return opcode.name() + " " + String.format("0x%01X", (f)) + ", "
+ String.format("0x%03X", (literal));
        }
    }
}

package com.vts.jplceditor.compiler.pic18;

/**
 *
 * @author Vusivus
 */
public enum Opcode {
    ADDWF((short) 6, (short) 0x24),
    ADDWFC((short) 6, (short) 0x20),
    ANDWF((short) 6, (short) 0x14),
    CLRF((short) 7, (short) 0x6A),
    COMF((short) 6, (short) 0x1C),
    CPFSEQ((short) 7, (short) 0x62),
    CPFSGT((short) 7, (short) 0x64),
    CPFSLT((short) 7, (short) 0x60),
    DECF((short) 6, (short) 0x04),
    DECFSZ((short) 6, (short) 0x2C),
    DCFSNZ((short) 6, (short) 0x4C),
    INCF((short) 6, (short) 0x28),
    INCFSZ((short) 6, (short) 0x3C),
    INFSNZ((short) 6, (short) 0x48),
    IORWF((short) 6, (short) 0x10),
    MOVF((short) 6, (short) 0x50),
    MOVFF((short) 8, (short) 0xCF),
    MOVWF((short) 7, (short) 0x6E),
    MULWF((short) 7, (short) 0x02),
    NEGF((short) 7, (short) 0x6C),
}

```

```

RLCF((short) 6, (short) 0x34),
RLNCF((short) 6, (short) 0x44),
RRCF((short) 6, (short) 0x30),
RRNCF((short) 6, (short) 0x40),
SETF((short) 7, (short) 0x68),
SUBFWB((short) 6, (short) 0x54),
SUBWF((short) 6, (short) 0x5C),
SUBWFB((short) 6, (short) 0x58),
SWAPF((short) 6, (short) 0x38),
TSTFSZ((short) 6, (short) 0x66),
XORWF((short) 6, (short) 0x18),

BCF((short) 4, (short) 0x90),
BSF((short) 4, (short) 0x80),
BTFSC((short) 4, (short) 0xB0),
BTFSS((short) 4, (short) 0xA0),
BTG((short) 4, (short) 0x70),

CALL((short) 11, (short) 0xECF),
BRA((short) 5, (short) 0xD0),
GOTO((short) 12, (short) 0xEFF),
NOP((short) 16, (short) 0x0),
RETURN((short) 15, (short) 0x12),

ADDLW((short) 8, (short) 0x0F),
ANDLW((short) 8, (short) 0x0B),
IORLW((short) 8, (short) 0x09),
LFSR((short) 10, (short) 0x0EE0),
MOVLB((short) 12, (short) 0x010),
MOVLW((short) 8, (short) 0x0E),
MULLW((short) 8, (short) 0xD0),
SUBLW((short) 8, (short) 0x08),
XORLW((short) 8, (short) 0x0A);

private final short bits;
private final short hexCode;

private Opcode(short bits, short hexCode) {
    this.bits = bits;
    this.hexCode = hexCode;
}

public short getBits() {
    return bits;
}

public short getHexCode() {
    return hexCode;
}

}

package com.vts.jplceditor.compiler.pic18;

import com.vts.jplceditor.compiler.pic18.mem.MemoryRange;

```

```
import com.vts.jplceditor.compiler.pic18.mem.Pic18f2550Ram;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

/**
 *
 * @author Vusivus
 */
public class Pic18 extends Pic18f2550Ram {

    public static final int W16AL = 0xB5;
    public static final int W16BL = 0xB7;
    public static final int W16CL = 0xB9;
    public static final int W16DL = 0xBB;
    public static final int W16AH = 0x8B6;
    public static final int W16BH = 0xB8;
    public static final int W16CH = 0xBA;
    public static final int W16DH = 0xBC;

    public static final int WREGA = 0xBC;
    public static final int WRGB = 0xBC;
    public static final int WREGC = 0xBC;
    public static final int WREGD = 0xBC;
    public static final int WREGE = 0xBC;

    public static final int W32_0 = 0xBD;
    public static final int W32_1 = 0xBE;
    public static final int W32_2 = 0xBF;
    public static final int W32_3 = 0xC0;
    public final int freeRamSlot = 0xC1;

    public static int CONFIG1L = 0x03;
    public static int CONFIG1H = 0x0C;
    public static int CONFIG2L = 0x36;
    public static int CONFIG2H = 0x13;
    public static int CONFIG3L = 0;
    public static int CONFIG3H = 0x85;
    public static int CONFIG4L = 0xA5;
    public static int CONFIG4H = 0;
    public static int CONFIG5L = 0x3F;
    public static int CONFIG5H = 0xC0;
    public static int CONFIG6L = 0;
    public static int CONFIG6H = 0;
    public static int CONFIG7L = 0;
    public static int CONFIG7H = 0;
    public static int DEVID2 = 0X12;
    public static int DEVID1 = 0X40;

    public static boolean w = false;
    public static boolean f = true;
    public static boolean bsr = true;
    public static boolean acc = false;

    private final HashMap<Integer, Pic18Instruction> picProgram;
    private int currentAddress, nextAddress;
```

```

        public Pic18(List<MemoryRange> gpRegisters, List<MemoryRange>
sfRegisters) {
            super(gpRegisters, sfRegisters);
            picProgram = new HashMap<>();
            currentAddress = 0;
            nextAddress = 1;
        }

        public void addInstruction(Pic18Instruction instruction) {
            picProgram.put(nextAddress, instruction);
            currentAddress += instruction.getWordCount();
            nextAddress += instruction.getWordCount();
        }

        public void write16Multiply() {
            //labels.add(new Subroutine("Multiply16", nextAddress));
            addInstruction(new LiteralInstruction(Opcode.MOVLW, 0));
            addInstruction(new ByteInstruction(Opcode.MOVWF, BSR, f, acc));

            addInstruction(new ByteInstruction(Opcode.MOVF, W16AL, w, bsr));
            addInstruction(new LiteralInstruction(Opcode.MULWF, W16BL));
            addInstruction(new ByteInstruction(Opcode.MOVFF, PRODH, f, acc));
            picProgram.get(currentAddress).setFile2(W32_1);
            addInstruction(new ByteInstruction(Opcode.MOVFF, PRODL, f, acc));
            picProgram.get(currentAddress).setFile2(W32_0);

            addInstruction(new ByteInstruction(Opcode.MOVF, W16AH, w, bsr));
            addInstruction(new LiteralInstruction(Opcode.MULWF, W16BH));
            addInstruction(new ByteInstruction(Opcode.MOVFF, PRODH, f, acc));
            picProgram.get(currentAddress).setFile2(W32_3);
            addInstruction(new ByteInstruction(Opcode.MOVFF, PRODL, f, acc));
            picProgram.get(currentAddress).setFile2(W32_2);

            addInstruction(new ByteInstruction(Opcode.MOVF, W16AL, w, bsr));
            addInstruction(new LiteralInstruction(Opcode.MULWF, W16BH));
            addInstruction(new ByteInstruction(Opcode.MOVF, PRODL, w, acc));
            addInstruction(new ByteInstruction(Opcode.ADDWF, W32_1, f, bsr));
            addInstruction(new ByteInstruction(Opcode.MOVF, PRODH, w, acc));
            addInstruction(new ByteInstruction(Opcode.ADDWFC, W32_2, f, bsr));
            addInstruction(new ByteInstruction(Opcode.CLRF, WREG, w, acc));
            addInstruction(new ByteInstruction(Opcode.ADDWFC, W32_3, f, bsr));
            addInstruction(new ControlInstruction(Opcode.RETURN, true));

        }

        public void write16Divide() {
            //labels.add(new Subroutine("Divide16", nextAddress));

            int dloop;
    }

```

```

int nochk;
int nogo;

addInstruction(new LiteralInstruction(Opcode.MOVLW, 0));
addInstruction(new ByteInstruction(Opcode.MOVWF, BSR, f, acc));
//First move the 16 bit register b to register d
addInstruction(new BitInstruction(Opcode.BCF, INTCON, (byte) 7,
acc));
addInstruction(new ByteInstruction(Opcode.MOVFF, W16BH, false, acc));
picProgram.get(currentAddress).setFile2(W16DH);
addInstruction(new ByteInstruction(Opcode.MOVFF, W16BL, false, acc));
picProgram.get(currentAddress).setFile2(W16DL);
addInstruction(new BitInstruction(Opcode.BSF, INTCON, (byte) 7,
acc));
// Clear register b and c
addInstruction(new ByteInstruction(Opcode.CLRF, W16BH, w, acc));
addInstruction(new ByteInstruction(Opcode.CLRF, W16BL, w, acc));
addInstruction(new ByteInstruction(Opcode.CLRF, W16CH, w, acc));
addInstruction(new ByteInstruction(Opcode.CLRF, W16CL, w, acc));
//Set up a loop counter
addInstruction(new LiteralInstruction(Opcode.MOVLW, 0xF));
addInstruction(new ByteInstruction(Opcode.MOVWF, WREGA, f, bsr));
dloop = nextAddress;
addInstruction(new BitInstruction(Opcode.BCF, STATUS, (byte) 0,
acc));
addInstruction(new ByteInstruction(Opcode.RLCF, W16DL, f, bsr));
addInstruction(new ByteInstruction(Opcode.RLCF, W16DH, f, bsr));
addInstruction(new ByteInstruction(Opcode.RLCF, W16CL, f, bsr));
addInstruction(new ByteInstruction(Opcode.RLCF, W16CH, f, bsr));
addInstruction(new ByteInstruction(Opcode.RLCF, W16DL, f, bsr));
addInstruction(new ByteInstruction(Opcode.MOVF, W16AH, w, bsr));
addInstruction(new ByteInstruction(Opcode.SUBWF, W16CH, w, bsr));
addInstruction(new BitInstruction(Opcode.BTFSS, STATUS, (byte) 2,
acc));
nochk = dloop + 0xB;
nogo = 0xA + nochk;
addInstruction(new ControlInstruction(Opcode.GOTO, nochk));
addInstruction(new ByteInstruction(Opcode.MOVF, W16AL, w, bsr));
addInstruction(new ByteInstruction(Opcode.SUBWF, W16CL, w, bsr));
//nochk subroutine
addInstruction(new BitInstruction(Opcode.BTFSS, STATUS, (byte) 0,
acc));
addInstruction(new ControlInstruction(Opcode.GOTO, nogeo));
addInstruction(new ByteInstruction(Opcode.MOVF, W16AL, w, bsr));
addInstruction(new ByteInstruction(Opcode.SUBWF, W16CL, f, bsr));
addInstruction(new BitInstruction(Opcode.BTFSS, STATUS, (byte) 0,
acc));
addInstruction(new ByteInstruction(Opcode.DECF, W16AH, f, bsr));
addInstruction(new ByteInstruction(Opcode.MOVF, W16AH, w, bsr));
addInstruction(new ByteInstruction(Opcode.SUBWF, W16CH, f, bsr));
addInstruction(new ByteInstruction(Opcode.SUBWF, W16CH, f, bsr));
addInstruction(new BitInstruction(Opcode.BSF, STATUS, (byte) 0,
acc));
//nogo subroutine
addInstruction(new ByteInstruction(Opcode.RLCF, W16BL, f, bsr));
addInstruction(new ByteInstruction(Opcode.RLCF, W16BH, f, bsr));

```

```
        addInstruction(new BitInstruction(Opcode.DECFSZ, WREGA, (byte) 0,
acc));
        addInstruction(new ControlInstruction(Opcode.GOTO, dloop));
        addInstruction(new ControlInstruction(Opcode.RETURN, true));
    }

    public void write16Add() {
    }

    public void write16Subtract() {
    }

    public void writeFloatAdd() {
    }

    public void writeFloatSubtract() {
    }

    public void writeFloatMultiply() {
    }

    public void writeFloatDivide() {
    }

    public void write16Compare() {
    }

    public void writeFloatCompare() {
    }
}
```

```
package com.vts.jplceditor.compiler.pic18;

import com.vts.jplceditor.compiler.pic18.hex.HexWriter;
import com.vts.jplceditor.compiler.pic18.mem.MemoryRange;
import com.vts.jplceditor.compiler.pic18.mem.Pic18f2550Ram;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.compiler.plc.PLCOOperator;
import com.vts.jplceditor.compiler.plc.instruction.PLCInstr;
import com.vts.jplceditor.compiler.plc.instruction.PLC_Add;
import com.vts.jplceditor.compiler.plc.instruction.PLC_And;
import com.vts.jplceditor.compiler.plc.instruction.PLC_Equ;
import com.vts.jplceditor.compiler.plc.instruction.PLC_Grt;
import com.vts.jplceditor.compiler.plc.instruction.PLC_Les;
import com.vts.jplceditor.compiler.plc.instruction.PLC_Orl;
import com.vts.jplceditor.compiler.plc.instruction.PLC_Ote;
```

```

import com.vts.jplceditor.compiler.plc.instruction.PLC_Otl;
import com.vts.jplceditor.compiler.plc.instruction.PLC_Otu;
import com.vts.jplceditor.compiler.plc.instruction.PLC_Rung;
import com.vts.jplceditor.compiler.plc.instruction.PLC_Sub;
import com.vts.jplceditor.compiler.plc.instruction.PLC_Xic;
import com.vts.jplceditor.compiler.plc.instruction.PLC_Xio;
import com.vts.jplceditor.compiler.plc.instruction.PLC_Xor;
import com.vts.jplceditor.compiler.plc.instruction.Plc_Mov;
import com.vts.jplceditor.compiler.plc.instruction.Plc_Mvm;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusivus
 */
public class Pic18Compiler extends Pic18 {

    private List<PLCInstr> plcInstructions;
    private List<Pic18Instruction> picInstructions;
    private int org = (int) Pic18Mem.PROGRAM_ORG; // where user program
starts
    private StringBuffer asmOutput, hexOutput;
    private Pic18Prog picProgram;
    private int address = org;
    private String adr;

    public Pic18Compiler(List<MemoryRange> gpRegisters, List<MemoryRange>
sfRegisters) {
        super(gpRegisters, sfRegisters);
    }

    public static Pic18Compiler forPic18f2550() {
        Pic18f2550Ram ram = Pic18f2550Ram.create();
        return new Pic18Compiler(ram.getGpRegisters(), ram.getSfRegisters());
    }

    public void compileInstructionList(List<PLC_Rung> list, List<PLCOperand>
tagList) {
        plcInstructions = new ArrayList<>();
        picInstructions = new ArrayList<>();
        verifyTypes(list, tagList); //assign data types to instruction
operands
        assignAddresses(list, tagList); //assign addresses to operands and
tags
        //PLC_Rung
        compilePLC(list); //generate a PLC instruction list
        compilePICAsm(); //generate a Pic instruction list
        compilePICHex(); // generate hex code
    }

    private void compilePICAsm() {
        asmOutput = new StringBuffer();
        asmOutput.append("org 0x2000\n");
        plcInstructions.stream().forEach((plci) -> {
            plci.getPicAsm().stream().map((picI) -> {
                picI.Assemble();
            })
        });
    }
}

```

```

        adr = String.format("0x%04X", address);
        asmOutput.append(adr + " : " + picI.toString() + "\n");
        address += (picI.getWordCount()) * 2;
        return picI;
    }).forEach((picI) -> {
        picInstructions.add(picI);
    });
}
ControlInstruction ret = new ControlInstruction(Opcode.RETURN,
false);
ret.Assemble();
picInstructions.add(ret);
asmOutput.append("RET");
}

private void compilePICHex() {
    picProgram = new Pic18Prog((int) Pic18Mem.PROGRAM_ORG,
picInstructions);
    HexWriter writer = new HexWriter(picProgram);
    writer.writeOS();
    writer.writeCode();
    writer.writeConfig();
    hexOutput = writer.getHexContent();
}

private void compilePLC(List<PLC_Rung> rungs) {
    rungs.stream().forEach((rung) -> { //Add each instruction to the asm
instruction list
        int size = rung.getInstructions().size(); //number of
instructions in rung
        PLCInstruction crrI;
        for (int i = 0; i < size; i++) {
            PLCOperand prevENO;
            crrI = rung.getInstructions().get(i);
            if (i != 0) {
                prevENO = rung.getInstructions().get(i -
1).getOperands().get(1);
            } else {
                prevENO = null;
            }
            addToAsmList(prevENO, crrI);
        }
    });
}

public void addToAsmList(PLCOperand prevENO, PLCInstruction crrI) {
    switch (crrI.getOperator()) {
        case EQU:
            PLC_Equ equ = new PLC_Equ(org, crrI.getOperands(), prevENO);
            plcInstructions.add(equ);
            org += equ.getwordCount() * 2;
            break;
        case LES:
            PLC_Les les = new PLC_Les(org, crrI.getOperands(), prevENO);
            plcInstructions.add(les);
            org += les.getwordCount() * 2;
    }
}

```

```

        break;
case GRT:
    PLC_Grt grt = new PLC_Grt(org, crrI.getOperands(), prevENO);
    plcInstructions.add(grt);
    org += grt.getwordCount() * 2;
    break;
case ADD:
    PLC_Add add = new PLC_Add(org, crrI.getOperands(), prevENO);
    plcInstructions.add(add);
    org += add.getwordCount() * 2;
    break;
case SUB:
    PLC_Sub sub = new PLC_Sub(org, crrI.getOperands(), prevENO);
    plcInstructions.add(sub);
    org += sub.getwordCount() * 2;
    break;
case XIC:
    PLC_Xic xic = new PLC_Xic(org, crrI.getOperands(), prevENO);
    plcInstructions.add(xic);
    org += xic.getwordCount() * 2;
    break;
case XIO:
    PLC_Xio xio = new PLC_Xio(org, crrI.getOperands(), prevENO);
    plcInstructions.add(xio);
    org += xio.getwordCount() * 2;
    break;
case OTE:
    PLC_Ote ote = new PLC_Ote(org, crrI.getOperands(), prevENO);
    plcInstructions.add(ote);
    org += ote.getwordCount() * 2;
    break;
case OTL:
    PLC_Otl otl = new PLC_Otl(org, crrI.getOperands(), prevENO);
    plcInstructions.add(otl);
    org += otl.getwordCount() * 2;
    break;
case OTU:
    PLC_Otu otu = new PLC_Otu(org, crrI.getOperands(), prevENO);
    plcInstructions.add(otu);
    org += otu.getwordCount() * 2;
    break;
case AND:
    PLC_And and = new PLC_And(org, crrI.getOperands(), prevENO);
    plcInstructions.add(and);
    org += and.getwordCount() * 2;
    break;
case OR:
    PLC_Or1 or = new PLC_Or1(org, crrI.getOperands(), prevENO);
    plcInstructions.add(or);
    org += or.getwordCount() * 2;
    break;
case XOR:
    PLC_Xor xor = new PLC_Xor(org, crrI.getOperands(), prevENO);
    plcInstructions.add(xor);
    org += xor.getwordCount() * 2;
    break;
case MOV:

```

```

        Plc_Mov mov = new Plc_Mov(org, crrI.getOperands(), prevENO);
        plcInstructions.add(mov);
        org += mov.getwordCount() * 2;
        break;
    case MVM:
        Plc_Mvm mvm = new Plc_Mvm(org, crrI.getOperands(), prevENO);
        plcInstructions.add(mvm);
        org += mvm.getwordCount() * 2;
        break;
    }
}

private void verifyTypes(List<PLC_Rung> list, List<PLCOperand> tagList) {
    list.stream().forEach((rung) -> {
        rung.getInstructions().stream().forEach((inst) -> {
            for (PLCOperand op : inst.getOperands()) {
                for (PLCOperand tag : tagList) {
                    if (op.getIdentity().equals(tag.getIdentity())) {
                        op.setDataType(tag.getDataType());
                        break;
                    }
                }
            }
        });
    });
}

private void assignAddresses(List<PLC_Rung> list, List<PLCOperand> tagList) {
    int ramAddress = 0x100; //available RAM starts from 0x75 on PLC to
0x3FF
    byte bitNo = 0;
    int nextAddr;
    PicAddress adr;
    //First assign addresses to the user tags
    for (PLCOperand op : tagList) {
        if (null != op.getDataType()) {
            switch (op.getDataType()) {
                case BIT:
                    adr = new PicAddress(ramAddress, bitNo);
                    op.setAddress(adr);
                    ++bitNo;
                    if (bitNo > 7) {
                        bitNo = 0;
                        ++ramAddress;
                    }
                    break;
                case BYTE:
                    ++ramAddress;
                    adr = new PicAddress(ramAddress);
                    op.setAddress(adr);
                    break;
                case INT:
                    ++ramAddress;
                    adr = new PicAddress(ramAddress);
                    op.setAddress(adr);
                    ramAddress += 2;
            }
        }
    }
}

```

```

        break;
    case FLOAT:
        ++ramAddress;
        adr = new PicAddress(ramAddress);
        op.setAddress(adr);
        ramAddress += 4;
        break;
    default:
        break;
    }
}
//Next assign addresses to PLC Instructions
++ramAddress;
for (PLC_Rung rung : list) {
    for (PLCInstruction i : rung.getInstructions()) {
        //First set addresses on any instruction using tags
        for (PLCOperand tagOp : tagList) { //Tag list
            for (PLCOperand op : i.getOperands()) { //get the
                operands of each instruction
                    //if the operand name is equal to the tag name, give
                    it the same address.
                    if (op.getIdentity().equals(tagOp.getIdentity())) {
                        op.setAddress(tagOp.getAddress());
                    }
                }
            }
        }
        //next assign addresses to the EN and ENO for each
        instruction
        i.getOperands().get(0).setAddress(new PicAddress(ramAddress,
        (byte) 0)); //EN bit
        i.getOperands().get(1).setAddress(new PicAddress(ramAddress,
        (byte) 1)); //ENO bit
        //Next assign addresses to any I/O operands if any
        i.getOperands().stream().filter((op) ->
        (isIO(op.getIdentity()))).forEach((op) -> {
            setIOAddr(op); //if I/O operand found, assign its default
            address
        });
        //next assign the addresses to the timer instructions if they
        exist
        if (i.getOperator() == PLCOperator.CTD || i.getOperator() ==
        PLCOperator.CTU
            || i.getOperator() == PLCOperator.TON) {
            //assign addresses to the bit flags
            i.getOperands().get(2).setAddress(new
            PicAddress(ramAddress, (byte) 2));
            i.getOperands().get(3).setAddress(new
            PicAddress(ramAddress, (byte) 3));

            ++ramAddress;
            i.getOperands().get(4).setAddress(new
            PicAddress(ramAddress));
            ++ramAddress;
            i.getOperands().get(5).setAddress(new
            PicAddress(ramAddress));
            ramAddress += 2;
        }
    }
}

```

```
i.getOperands().get(6).setAddress(new  
PicAddress(ramAddress));  
}  
    ++ramAddress;  
}  
}  
  
public StringBuffer getAsmOutput() {  
    return asmOutput;  
}  
  
public StringBuffer getHexOutput() {  
    return hexOutput;  
}  
  
private boolean isIO(String identity) {  
    switch (identity) {  
        case "DIO":  
            return true;  
        case "DI1":  
            return true;  
        case "DI2":  
            return true;  
        case "DI3":  
            return true;  
        case "DI4":  
            return true;  
        case "DI5":  
            return true;  
        case "DI6":  
            return true;  
        case "DI7":  
            return true;  
        case "DO0":  
            return true;  
        case "DO1":  
            return true;  
        case "DO2":  
            return true;  
        case "DO3":  
            return true;  
        case "DO4":  
            return true;  
        case "DO5":  
            return true;  
        case "DO6":  
            return true;  
        case "DO7":  
            return true;  
        case "AI0":  
            return true;  
        case "AI1":  
            return true;  
        case "AI2":  
            return true;  
        case "AI3":  
            return true;
```

```
        return true;
    case "AI4":
        return true;
    case "AI5":
        return true;
    case "AI6":
        return true;
    case "AI7":
        return true;
    case "AO0":
        return true;
    default:
        return false;
    }
}

private void setIOAddr(PLCOperand op) {
    String identity = op.getIdentity();
    switch (identity) {
        case "DI0":
            op.setAddress(new PicAddress(0x70, (byte) 0));
            break;
        case "DI1":
            op.setAddress(new PicAddress(0x70, (byte) 1));
            break;
        case "DI2":
            op.setAddress(new PicAddress(0x70, (byte) 2));
            break;
        case "DI3":
            op.setAddress(new PicAddress(0x70, (byte) 3));
            break;
        case "DI4":
            op.setAddress(new PicAddress(0x70, (byte) 4));
            break;
        case "DI5":
            op.setAddress(new PicAddress(0x70, (byte) 5));
            break;
        case "DI6":
            op.setAddress(new PicAddress(0x70, (byte) 6));
            break;
        case "DI7":
            op.setAddress(new PicAddress(0x70, (byte) 7));
            break;
        case "DO0":
            op.setAddress(new PicAddress(0x73, (byte) 0));
            break;
        case "DO1":
            op.setAddress(new PicAddress(0x73, (byte) 1));
            break;
        case "DO2":
            op.setAddress(new PicAddress(0x73, (byte) 2));
            break;
        case "DO3":
            op.setAddress(new PicAddress(0x73, (byte) 3));
            break;
        case "DO4":
            op.setAddress(new PicAddress(0x73, (byte) 4));
            break;
    }
}
```

```

        break;
    case "D05":
        op.setAddress(new PicAddress(0x73, (byte) 5));
        break;
    case "D06":
        op.setAddress(new PicAddress(0x73, (byte) 6));
        break;
    case "D07":
        op.setAddress(new PicAddress(0x73, (byte) 7));
        break;
    case "AI0":
        op.setAddress(new PicAddress(0x60));
        break;
    case "AI1":
        op.setAddress(new PicAddress(0x62));
        break;
    case "AI2":
        op.setAddress(new PicAddress(0x64));
        break;
    case "AI3":
        op.setAddress(new PicAddress(0x66));
        break;
    case "AI4":
        op.setAddress(new PicAddress(0x68));
        break;
    case "AI5":
        op.setAddress(new PicAddress(0x6A));
        break;
    case "AI6":
        op.setAddress(new PicAddress(0x6C));
        break;
    case "AI7":
        op.setAddress(new PicAddress(0x6E));
        break;
    case "AO0":
        op.setAddress(new PicAddress(0x71));
        break;
    }
}

public Pic18Prog getPicProgram() {
    return picProgram;
}

}

```

```

package com.vts.jplceditor.compiler.pic18;

/**
 *
 * @author Vusivus
 */
public interface Pic18Instruction {
    public void Assemble();
}

```

```
public short get16BitCode();
public short getSecond16BitCode();
public int getWordCount();
public void setFile(int file);
public void setFile2(int file);
}

package com.vts.jplceditor.compiler.pic18;

/**
 *
 * @author Vusman
 */
public abstract class Pic18Mem {

    public int TOSU = 0xFFFF;
    public int INDF21 = 0xFDF;
    public int CCPR1H = 0xFB0;
    public int IPR1 = 0xF9F;
    public int TOSH = 0xFFE;
    public int POSTINC2 = 0xFDE;
    public int CCPR1L = 0xFB0;
    public int PIR1 = 0xF9E;
    public int UEP14 = 0xF7E;
    public int TOSL = 0xFFD;
    public int POSTDEC2 = 0xFDD;
    public int CCP1CON = 0xBD;
    public int PIE1 = 0xF9D;
    public int UEP13 = 0xF7D;
    public int STKPTR = 0xFFC;
    public int PREINC2 = 0xFDC;
    public int CCPR2H = 0xFBC;
    public int UEP12 = 0xF7C;
    public int PCLATU = 0xFFB;
    public int PLUSW2 = 0xFDB;
    public int CCPR2L = 0xFBB;
    public int OSCTUNE = 0xF9B;
    public int UEP11 = 0xF7B;
    public int PCLATH = 0xFFA;
    public int FSR2H = 0xFD0;
    public int CCP2CON = 0xFBA;
    public int UEP10 = 0xF7A;
    public int PCL = 0xFF9;
    public int FSR2L = 0xFD9;
    public int UEP9 = 0xF79;
    public int TBLPTRU = 0xFF8;
    public int STATUS = 0xFD8;
    public int BAUDCON = 0xFB8;
    public int UEP8 = 0xF78;
    public int TBLPTRH = 0xFF7;
    public int TMROH = 0xFD7;
    public int ECCP1DEL = 0xFB7;
    public int UEP7 = 0xF77;
    public int TBLPTRL = 0xFF6;
```

```
public int TMR0L = 0xFD6;
public int ECCP1AS = 0xFB6;
public int TRISE = 0xF96;
public int UEP6 = 0xF76;
public int TABLAT = 0xFF5;
public int T0CON = 0xFD5;
public int CVRCON = 0xFB5;
public int TRISD = 0xF95;
public int UEP5 = 0xF75;
public int PRODH = 0xFF4;
public int CMCON = 0xFB4;
public int TRISC = 0xF94;
public int UEP4 = 0xF74;
public int PRODL = 0xFF3;
public int OSCCON = 0xFD3;
public int TMR3H = 0xFB3;
public int TRISB = 0xF93;
public int UEP3 = 0xF73;
public int INTCON = 0xFF2;
public int HLVDCON = 0xFD2;
public int TMR3L = 0xFB2;
public int TRISA = 0xF92;
public int UEP2 = 0xF72;
public int INTCON2 = 0xFF1;
public int WDTCON = 0xFD1;
public int T3CON = 0xFB1;
public int UEP1 = 0xF71;
public int INTCON3 = 0xFF0;
public int RCON = 0xFD0;
public int SPBRGH = 0xFB0;
public int UEP0 = 0xF70;
public int INDF0 = 0xFEF;
public int TMR1H = 0xFCF;
public int SPBRG = 0xFAF;
public int UCFG = 0xF6F;
public int POSTINCO = 0xFEE;
public int TMR1L = 0xFCE;
public int RCREG = 0xFAE;
public int UADDR = 0xF6E;
public int POSTDEC0 = 0xFED;
public int T1CON = 0xFCD;
public int TXREG = 0xFAD;
public int LATE = 0xF8D;
public int UCON = 0xF6D;
public int PREINCO = 0xFEC;
public int TMR2 = 0xFCC;
public int TXSTA = 0xFAC;
public int LATD = 0xF8C;
public int USTAT = 0xF6C;
public int PLUSW = 0xFEB;
public int PR2 = 0xFCB;
public int RCSTA = 0xFAB;
public int LATC = 0xF8B;
public int UEIE = 0xF6B;
public int FSR0H = 0xEA;
public int T2CON = 0xFCA;
public int LATB = 0xF8A;
```

```
public int UEIR = 0xF6A;
public int FSR0L = 0xFE9;
public int SSPBUF = 0xFC9;
public int EEADR = 0xFA9;
public int LATA = 0xF89;
public int UIE = 0xF69;
public int WREG = 0xFE8;
public int SSPADD = 0xFC8;
public int EEDATA = 0xFA8;
public int UIR = 0xF68;
public int INDF1 = 0xFE7;
public int SSPSTAT = 0xFC7;
public int EECON2 = 0xFA7;
public int UFRMH = 0xF67;
public int POSTINC1 = 0xFE6;
public int SSPCON1 = 0xFC6;
public int EECON1 = 0xFA6;
public int UFRML = 0xF66;
public int POSTDEC1 = 0xFE5;
public int SSPCON2 = 0xFC5;
public int SPPCON = 0xF65;
public int PREINC1 = 0xFE4;
public int ADRESH = 0xFC4;
public int PORTE = 0xF84;
public int SPPEPS = 0xF64;
public int PLUSW1 = 0xFE3;
public int ADRESL = 0xFC3;
public int PORTD = 0xF83;
public int SPPCFG = 0xF63;
public int FSR1H = 0xFE2;
public int ADCON0 = 0xFC2;
public int IPR2 = 0xFA2;
public int PORTC = 0xF82;
public int SPPDATA = 0xF62;
public int FSR1L = 0xFE1;
public int ADCON1 = 0xFC1;
public int PIR2 = 0xFA1;
public int PORTB = 0xF81;
public int BSR = 0xFE0;
public int ADCON2 = 0xFC0;
public int PIE2 = 0xFA0;
public int PORTA = 0xF80;
public final boolean w = false;
public final boolean f = true;
public final boolean a = false;
public final boolean bsr = true;
public final byte C = 0;
public final byte DC = 1;
public final byte Z = 2;
public final byte OV = 3;
public final byte N = 4;

public static int CONFIG1L = 0x0;
public static int CONFIG1H = 0x0C;
public static int CONFIG2L = 0x1F;
public static int CONFIG2H = 0x1F;
public static int CONFIG3L = 0;
```

```
public static int CONFIG3H = 0x83;
public static int CONFIG4L = 0x81;
public static int CONFIG4H = 0;
public static int CONFIG5L = 0xF;
public static int CONFIG5H = 0xC0;
public static int CONFIG6L = 0x0F;
public static int CONFIG6H = 0;
public static int CONFIG7L = 0;
public static int CONFIG7H = 0;
public static int DEVID2 = 0X12;
public static int DEVID1 = 0X40;

public static long PROGRAM_ORG = 8192;
public static short IO_START = 0x60;
public static int BANK1 = 0x100;
public static int BANK2 = 0x200;
}

package com.vts.jplceditor.compiler.pic18;

import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.util.List;

/**
 *
 * @author Vusman
 */
public class Pic18Prog {

    private ByteBuffer codeHex;
    private ByteBuffer configHex;
    private ByteBuffer devIdHex;
    private ByteOrder byteOrder = ByteOrder.LITTLE_ENDIAN;
    private int size = 0;
    private final int org;
    private List<Pic18Instruction> program;

    public Pic18Prog(int org, List<Pic18Instruction> prog) {
        this.org = org;
        program = prog;
        program.stream().forEach((i) -> {
            size += i.getWordCount();
        });
        System.out.println("Buffer Size is " + size);
        size *= 2;
        codeHex = ByteBuffer.allocate(size);
        codeHex.order(byteOrder);
        putProg(program);
        configHex = ByteBuffer.allocate(14);
        configHex.order(byteOrder);
        putConfig();
        devIdHex = ByteBuffer.allocate(2);
        putDevId();
    }
}
```

```
private void putProg(List<Pic18Instruction> prog) {
    prog.stream().forEach((i) -> {
        if (i.getWordCount() == 2) {
            codeHex.putShort(i.get16BitCode());
            codeHex.putShort(i.getSecond16BitCode());
        } else {
            codeHex.putShort(i.get16BitCode());
        }
    });
    //fixBytes(codeHex);
    codeHex.rewind();
}

public byte readCodeByte(int index) {
    return codeHex.get(index);
}

public int codeBufferPosition() {
    return codeHex.position();
}

public ByteBuffer getCodeHex() {
    return codeHex;
}

public byte readConfigByte() {
    return configHex.get();
}

public ByteBuffer getConfigHex() {
    return configHex;
}

private void putConfig() {
    configHex.put((byte) Pic18Mem.CONFIG1H);
    configHex.put((byte) Pic18Mem.CONFIG1L);
    configHex.put((byte) Pic18Mem.CONFIG2H);
    configHex.put((byte) Pic18Mem.CONFIG2L);
    configHex.put((byte) Pic18Mem.CONFIG3H);
    configHex.put((byte) Pic18Mem.CONFIG3L);
    configHex.put((byte) Pic18Mem.CONFIG4H);
    configHex.put((byte) Pic18Mem.CONFIG4L);
    configHex.put((byte) Pic18Mem.CONFIG5H);
    configHex.put((byte) Pic18Mem.CONFIG5L);
    configHex.put((byte) Pic18Mem.CONFIG6H);
    configHex.put((byte) Pic18Mem.CONFIG6L);
    configHex.put((byte) Pic18Mem.CONFIG7H);
    configHex.put((byte) Pic18Mem.CONFIG7L);
    fixBytes(configHex);
    configHex.rewind();
}

private void putDevId() {
    devIdHex.put((byte) Pic18Mem.DEVID2);
    devIdHex.put((byte) Pic18Mem.DEVID1);
    fixBytes(devIdHex);
    devIdHex.rewind();
}
```

```
}

private void fixBytes(ByteBuffer buf) {
    for (int i = 0; i < buf.capacity(); i++) {
        byte b = swapNibbles(buf.get(i));
        buf.put(i, b);
    }
}

private byte swapNibbles(byte x) {
    return (byte) ((x & 0x0F) << 4 | (x & 0xF0) >> 4);
}

public ByteBuffer getDevIdHex() {
    return devIdHex;
}

public int getSize() {
    return size;
}

public int getOrg() {
    return org;
}

public StringBuffer getProgramMap() {
    StringBuffer buff = new StringBuffer();
    for (int i = 0; i < codeHex.capacity(); i++) {
        buff.append(String.format("0x%04X", org + i)).append(" : ");
        buff.append(String.format("0x%02X", codeHex.get(i)));
        buff.append("\n");
    }
    return buff;
}

public String getProgramInformation() {
    StringBuilder info = new StringBuilder();
    info.append("=====\n");
    info.append("\nProgram Start : ").append(String.format("0x%04X", org));
    info.append("\n");
    info.append("Program Size : ").append(codeHex.capacity()).append(" bytes.");
    info.append("\n");

    return info.toString();
}

}

package com.vts.jplceditor.compiler.pic18;
```

```
/*
 *
 * @author Vusivus
 */
public class PicAddress {

    private int fileAddr;
    private byte bitNo;
    private boolean isBit;

    public PicAddress(int address) {
        this.fileAddr = address;
        isBit = false;
    }

    public PicAddress(int address, byte bitNo) {
        this.fileAddr = address;
        this.bitNo = bitNo;
        isBit = true;
    }

    public int getFileAddr() {
        return fileAddr;
    }

    public byte getBitNo() {
        return bitNo;
    }

    @Override
    public String toString() {
        if (isBit) {
            return String.format("0x%03X", fileAddr) + ", BitNumber:" +
bitNo;
        } else {
            return String.format("0x%03X", fileAddr);
        }
    }
}
```

```
package com.vts.jplceditor.compiler.pic18;
```

```
/*
 *
 * @author Vusivus
 */
public class PicAddress {

    private int fileAddr;
    private byte bitNo;
    private boolean isBit;

    public PicAddress(int address) {
        this.fileAddr = address;
    }
```

```
    isBit = false;
}

public PicAddress(int address, byte bitNo) {
    this.fileAddr = address;
    this.bitNo = bitNo;
    isBit = true;
}

public int getFileAddr() {
    return fileAddr;
}

public byte getBitNo() {
    return bitNo;
}

@Override
public String toString() {
    if (isBit) {
        return String.format("0x%03X", fileAddr) + ", BitNumber:" +
bitNo;
    } else {
        return String.format("0x%03X", fileAddr);
    }
}
}

package com.vts.jplceditor.compiler.pic18;

/**
 *
 * @author Vusivus
 */
public class PicAddress {

    private int fileAddr;
    private byte bitNo;
    private boolean isBit;

    public PicAddress(int address) {
        this.fileAddr = address;
        isBit = false;
    }

    public PicAddress(int address, byte bitNo) {
        this.fileAddr = address;
        this.bitNo = bitNo;
        isBit = true;
    }

    public int getFileAddr() {
        return fileAddr;
    }

    public byte getBitNo() {
```

```

        return bitNo;
    }

    @Override
    public String toString() {
        if (isBit) {
            return String.format("0x%03X", fileAddr) + ", BitNumber:" +
bitNo;
        } else {
            return String.format("0x%03X", fileAddr);
        }
    }
}

package com.vts.jplceditor.compiler.pic18.hex;

/**
 *
 * @author Vusivus
 */
public class ExtendedLinearAddressRecord extends HexRecord{

    private final int upperAddress;
    public ExtendedLinearAddressRecord(byte[] bytes) {
        super(2, 0, bytes, RecordType.EXT_LINEAR_ADDR);

        upperAddress = ((bytes[0] & 0x00ff) << 24) + ((bytes[1] & 0x00ff) <<
16);
    }

    public ExtendedLinearAddressRecord(int length, int address, byte[] bytes)
{
        super(length, address, bytes, RecordType.EXT_LINEAR_ADDR);
        if (length != 2) {
            throw new IllegalArgumentException("length of a segment record
must be two, but is " + length);
        }
        if (address != 0) {
            throw new IllegalArgumentException("address of a segment record
must be zero, but is " + length);
        }
        upperAddress = ((bytes[0] & 0x00ff) << 24) + ((bytes[1] & 0x00ff) <<
16);
    }

    public int getUpperAddress() {
        return upperAddress;
    }
}
package com.vts.jplceditor.compiler.pic18.hex;

```

```
/**
 *
 * @author Vusivus
 */
public class ExtendedSegmentAddressRecord extends HexRecord {

    private final int segment;

    public ExtendedSegmentAddressRecord(byte[] bytes) {
        super(2, 0, bytes, RecordType.EXT_SEG_ADDR);

        segment = ((bytes[0] & 0x00ff) << 8) + (bytes[1] & 0x00ff);
    }

    public ExtendedSegmentAddressRecord(int length, int address, byte[]
bytes) {
        super(length, address, bytes, RecordType.EXT_SEG_ADDR);
        if (length != 2) {
            throw new IllegalArgumentException("length of a segment record
must be two, but is " + length);
        }
        if (address != 0) {
            throw new IllegalArgumentException("address of a segment record
must be zero, but is " + length);
        }
        segment = ((bytes[0] & 0x00ff) << 8) + (bytes[1] & 0x00ff);
    }

    public int getSegment() {
        return segment;
    }

    public int getUpperAddress() {
        return segment << 8;
    }
}
```

```
package com.vts.jplceditor.compiler.pic18.hex;

/**
 *
 * @author Vusivus
 */
public class ExtendedSegmentAddressRecord extends HexRecord {

    private final int segment;

    public ExtendedSegmentAddressRecord(byte[] bytes) {
        super(2, 0, bytes, RecordType.EXT_SEG_ADDR);

        segment = ((bytes[0] & 0x00ff) << 8) + (bytes[1] & 0x00ff);
    }
```

```

        public ExtendedSegmentAddressRecord(int length, int address, byte[] bytes) {
            super(length, address, bytes, RecordType.EXT_SEG_ADDR);
            if (length != 2) {
                throw new IllegalArgumentException("length of a segment record must be two, but is " + length);
            }
            if (address != 0) {
                throw new IllegalArgumentException("address of a segment record must be zero, but is " + length);
            }
            segment = ((bytes[0] & 0x00ff) << 8) + (bytes[1] & 0x00ff);
        }

        public int getSegment() {
            return segment;
        }

        public int getUpperAddress() {
            return segment << 8;
        }
    }
}

```

```

package com.vts.jplceditor.compiler.pic18.hex;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class HexRecordParser {
    public static final Pattern HEXRECORD_PATTERN = Pattern.compile(":([a-fA-F0-9]{2})([a-fA-F0-9]{4})([a-fA-F0-9]{2})(([a-fA-F0-9]{2})*)([a-fA-F0-9]{2})");

    @SuppressWarnings("unchecked")
    public static <T extends HexRecord> T parse(final String line) {
        final Matcher recordMatcher =
HexRecordParser.HEXRECORD_PATTERN.matcher(line);
        if (!recordMatcher.matches()) {
            throw new IllegalArgumentException("wrong format " + line);
        }
        HexRecordParser.validateChecksum(line);
        final int length = Integer.parseInt(recordMatcher.group(1), 16);
        final int address = Integer.parseInt(recordMatcher.group(2), 16);
        final RecordType recordType =
RecordType.values()[Integer.parseInt(recordMatcher.group(3), 16)];
        final String data = recordMatcher.group(4);
        final byte[] bytes = new byte[length];
        for (int i = 0; i < bytes.length; i++) {
            bytes[i] = (byte) Integer.parseInt(data.substring(2 * i, 2 * i +
2), 16);
        }
        switch (recordType) {
        case EOF:
            return (T) new EOFRecord();
        }
    }
}

```

```

        case DATA:
            return (T) new DataRecord(length, address, bytes);
        case EXT_SEG_ADDR:
            return (T) new ExtendedSegmentAddressRecord(length, address,
bytes);
        case START_SEG_ADDR:
            return (T) new StartSegmentAddressRecord(length, address, bytes);
        case EXT_LINEAR_ADDR:
            return (T) new ExtendedLinearAddressRecord(length, address,
bytes);
        case START_LINEAR_ADDR:
            return (T) new StartLinearRecord(length, address, bytes);
        default:
            throw new IllegalStateException("dont know record type " +
recordType);
        }
    }

    private static void validateChecksum(final String line) {
        int checksum = 0;
        for (int i = 1; i + 1 < line.length(); i += 2) {
            final int byteValue = Integer.parseInt(line.substring(i, i + 2),
16);
            checksum += byteValue;
        }
        if ((checksum & 0x00ff) != 0) {
            throw new IllegalArgumentException("wrong checksum: " + (checksum
& 0x0ff) + " in line " + line);
        }
    }
}

package com.vts.jplceditor.compiler.pic18.hex;

import com.vts.jplceditor.compiler.pic18.Pic18Prog;

/**
 *
 * @author Vusivus
 */
public class HexWriter {

    public static final int MAX_LINE_LEN = 0x10;
    private final Pic18Prog program;
    private final StringBuffer hexContent;

    public HexWriter(Pic18Prog program) {
        this.program = program;
        hexContent = new StringBuffer();
    }

    public StringBuffer getHexContent() {
        return hexContent;
    }

    private int upper(final int address) {

```

```

        return (address >> 16) & 0xffff;
    }

    private void writeEOF() {
        writeLine(0, 0, 1, "", 0);
    }

    private void writeLine(final int len, final int address, final int type,
final String value, final int checksum) {
        hexContent.append(": " + to2hex(len) + to4hex(address) + to2hex(type)
+ value
                + to2hex(-len - (address >> 8) - address - type - checksum) +
"\n");
    }

    private String to2hex(final int value) {
        return String.format("%02x", value & 0xff);
    }

    private String to4hex(final int value) {
        return String.format("%04x", value & 0xffff);
    }

    public void writeOS() {
        hexContent.append(":101D16001AEC0FF0B886B06ACF0EAF6EAC8413DD46\n"
                + ":101D2600020E0B6E040E0C6EBA0E0D6E0D2EFED745\n"
                + ":101D36000C2EFCD70B2EFAD700000F0EC112070E81\n"
                + ":101D4600B4129280516A94909494816A936A8290B4\n"
                + ":101D56008294736B706B7F6B78DF010EE56F1DEC01\n"
                + ":101D660009F00C0EE56F1DEC09F080A002D0F3DE41\n"
                + ":081D760001D012DEFAD7FFD7FD\n"
                + ":101E340010EE29F0270E006E010E016EA70EF66E4D\n"
                + ":101E44001D0EF76E000EF86ED2EC0BF010EE80F063\n"
                + ":101E5400110E006E010E016E010EF66E1E0EF76E6F\n"
                + ":0A1E6400000EF86ED2EC0BF0120035\n"
                + ":101D7E00090229000101008032090400000203005B\n"
                + ":101D8E000000092101010001222100070581034005\n"
                + ":091D9E00000107050103400001EA\n"
                + ":091DA70052554E204D4F444500F9\n"
                + ":0D1DB00050524F4752414D204D4F444500C9\n"
                + ":101DBD0050524F4752414D4D494E472E2E2E2EED\n"
                + ":011DCD000015\n"
                + ":101DCE000600FF0901A10119012940150026FF0097\n"
                + ":101DDE0075089540810219012940750895409102B8\n"
                + ":011DEE00C034\n"
                + ":101DEF0012010002000000834120100010001027C\n"
                + ":021DFF000001E1\n"
                + ":101E0100434F4E4E454354454420544F2050432E9A\n"
                + ":011E110000D0\n"
                + ":101E120010035600540020004C0061006200730061\n"
                + ":0E1E22000E0356005400200050004C004300F8\n"
                + ":041E3000040309049A\n");
    }

    public void writeConfig() {
        hexContent.append(":020000040030CA\n"
                + ":0E000000210C391EFF8081FF0FC00FE00F4062\n"

```

```
        + ":00000001FF");
    }

public void writeCode() {
    int address = program.getOrg();
    int size = program.getSize();
    int pointer = 0;
    while (size > 0) {
        final StringBuilder value = new StringBuilder();
        int checksum = 0;
        int length = 0;

        if (size >= MAX_LINE_LEN) {
            size -= MAX_LINE_LEN;
            length = MAX_LINE_LEN;
        } else {
            length = size;
            size = 0;
        }
        for (int idx = 0; idx < length; idx++) {
            final int b = program.readCodeByte(pointer);
            value.append(to2hex(b));
            checksum += b;
            ++pointer;
        }
        if (length > 0) {
            writeLine(length, address, 0, value.toString(), checksum);
        }
        address += length;
    }
}

private void writeConfigMem() {
    writeLine(2, 0, 4, "0030", 0xCA);
    final StringBuilder value = new StringBuilder();
    int checksum = 0;
    for (int idx = 0; idx < 14; idx++) {
        final int b = program.readConfigByte();
        value.append(to2hex(b));
        checksum += b;
    }
    writeLine(14, 0x300000, 0, value.toString(), checksum);
}

private void writeDevId() {

}

package com.vts.jplceditor.compiler.pic18.hex;

/**
 *
 * @author Vusivus
```

```

*/
public enum RecordType {
    DATA(0),
    EOF(1),
    EXT_SEG_ADDR(2),
    START_SEG_ADDR(3),
    EXT_LINEAR_ADDR(4),
    START_LINEAR_ADDR(5);

    private final int code;

    private RecordType(int code) {
        this.code = code;
    }

    public int getCode() {
        return code;
    }
}

package com.vts.jplceditor.compiler.pic18.hex;

/**
 *
 * @author Vusivus
 */
public class StartLinearRecord extends HexRecord {

    int linearAddress;

    public StartLinearRecord(byte[] bytes) {
        super(4, 0, bytes, RecordType.START_LINEAR_ADDR);

        linearAddress = ((bytes[0] & 0x00ff) << 24) + ((bytes[1] & 0x00ff) <<
16) + ((bytes[2] & 0x00ff) << 8)
                + (bytes[3] & 0x00ff);
    }

    public StartLinearRecord(int length, int address, byte[] bytes) {
        super(length, address, bytes, RecordType.START_LINEAR_ADDR);
        if (length != 4) {
            throw new IllegalArgumentException("length of a segment start
record must be four, but is " + length);
        }
        if (address != 0) {
            throw new IllegalArgumentException("address of a segment record
must be zero, but is " + length);
        }
        linearAddress = ((bytes[0] & 0x00ff) << 24) + ((bytes[1] & 0x00ff) <<
16) + ((bytes[2] & 0x00ff) << 8)
                + (bytes[3] & 0x00ff);
    }

    public int getLinearAddress() {
        return linearAddress;
    }
}

```

```
}

package com.vts.jplceditor.compiler.pic18.hex;

/**
 *
 * @author Vusivus
 */
public class StartSegmentAddressRecord extends HexRecord {

    private final int segment;
    private final int offset;

    public StartSegmentAddressRecord(byte[] bytes) {
        super(4, 0, bytes, RecordType.START_SEG_ADDR);

        segment = ((bytes[0] & 0x00ff) << 8) + (bytes[1] & 0x00ff);
        offset = ((bytes[2] & 0x00ff) << 8) + (bytes[3] & 0x00ff);
    }

    public StartSegmentAddressRecord(int length, int address, byte[] bytes) {
        super(length, address, bytes, RecordType.START_SEG_ADDR);
        if (length != 4) {
            throw new IllegalArgumentException("length of a segment start record must be four, but is " + length);
        }
        if (address != 0) {
            throw new IllegalArgumentException("address of a segment record must be zero, but is " + address);
        }
        segment = ((bytes[0] & 0x00ff) << 8) + (bytes[1] & 0x00ff);
        offset = ((bytes[2] & 0x00ff) << 8) + (bytes[3] & 0x00ff);
    }

    public int getSegment() {
        return segment;
    }

    public int getOffset() {
        return offset;
    }
}
```

```
package com.vts.jplceditor.compiler.pic18.mem;

/**
 *
 * @author Vusivus
 */
public class MemoryRange {
```

```
private final int startAddress;
private final int endAddress;

public MemoryRange(int startAddress, int endAddress) {
    this.startAddress = startAddress;
    this.endAddress = endAddress;
}

public int getStartAddress() {
    return startAddress;
}

public int getEndAddress() {
    return endAddress;
}

public boolean contains(final int address) {
    return startAddress <= address && address <= endAddress;
}

}

package com.vts.jplceditor.compiler.pic18.mem;

/**
 *
 * @author Vusivus
 */
public class Pic18Config {

    public static int CONFIG1L = 0x03;
    public static int CONFIG1H = 0x0C;
    public static int CONFIG2L = 0x36;
    public static int CONFIG2H = 0x13;
    public static int CONFIG3L = 0;
    public static int CONFIG3H = 0x85;
    public static int CONFIG4L = 0xA5;
    public static int CONFIG4H = 0;
    public static int CONFIG5L = 0x3F;
    public static int CONFIG5H = 0xC0;
    public static int CONFIG6L = 0;
    public static int CONFIG6H = 0;
    public static int CONFIG7L = 0;
    public static int CONFIG7H = 0;
    public static int DEVID2 = 0X12;
    public static int DEVID1 = 0X40;
}

package com.vts.jplceditor.compiler.pic18.mem;

import java.util.ArrayList;
import java.util.List;
```

```
/*
 *
 * @author Vusivus
 */
public class Pic18f2550Ram extends RamMemoryModel {

    public int TOSU = 0xFFFF;
    public int INDF21 = 0xFDF;
    public int CCPR1H = 0xFB;
    public int IPR1 = 0xF9F;
    public int TOSH = 0xFFE;
    public int POSTINC2 = 0xFDE;
    public int CCPR1L = 0xFBE;
    public int PIR1 = 0xF9E;
    public int UEP14 = 0xF7E;
    public int TOSL = 0xFFD;
    public int POSTDEC2 = 0xFDD;
    public int CCP1CON = 0xFBD;
    public int PIE1 = 0xF9D;
    public int UEP13 = 0xF7D;
    public int STKPTR = 0xFFC;
    public int PREINC2 = 0xFDC;
    public int CCPR2H = 0xFBC;
    public int UEP12 = 0xF7C;
    public int PCLATU = 0xFFB;
    public int PLUSW2 = 0xFDB;
    public int CCPR2L = 0xFBB;
    public int OSCTUNE = 0xF9B;
    public int UEP11 = 0xF7B;
    public int PCLATH = 0xFFA;
    public int FSR2H = 0xFD;
    public int CCP2CON = 0xFBA;
    public int UEP10 = 0xF7A;
    public int PCL = 0xFF9;
    public int FSR2L = 0xFD9;
    public int UEP9 = 0xF79;
    public int TBLPTRU = 0xFF8;
    public int STATUS = 0xFD8;
    public int BAUDCON = 0xFB8;
    public int UEP8 = 0xF78;
    public int TBLPTRH = 0xFF7;
    public int TMR0H = 0xFD7;
    public int ECCP1DEL = 0xFB7;
    public int UEP7 = 0xF77;
    public int TBLPTRL = 0xFF6;
    public int TMR0L = 0xFD6;
    public int ECCP1AS = 0xFB6;
    public int TRISE = 0xF96;
    public int UEP6 = 0xF76;
    public int TABLAT = 0xFF5;
    public int T0CON = 0xFD5;
    public int CVRCON = 0xFB5;
    public int TRISD = 0xF95;
    public int UEP5 = 0xF75;
    public int PRODH = 0xFF4;
    public int CMCON = 0xFB4;
```

```
public int TRISC = 0xF94;
public int UEP4 = 0xF74;
public int PRODL = 0xFF3;
public int OSCCON = 0xFD3;
public int TMR3H = 0xFB3;
public int TRISB = 0xF93;
public int UEP3 = 0xF73;
public int INTCON = 0xFF2;
public int HLVDCON = 0xFD2;
public int TMR3L = 0xFB2;
public int TRISA = 0xF92;
public int UEP2 = 0xF72;
public int INTCON2 = 0xFF1;
public int WDTCON = 0xFD1;
public int T3CON = 0xFB1;
public int UEP1 = 0xF71;
public int INTCON3 = 0xFF0;
public int RCON = 0xFD0;
public int SPBRGH = 0xFB0;
public int UEPO = 0xF70;
public int INDF0 = 0xFEF;
public int TMR1H = 0xFCF;
public int SPBRG = 0xFAF;
public int UCFG = 0xF6F;
public int POSTINCO = 0xFEE;
public int TMR1L = 0xFCE;
public int RCREG = 0xFAE;
public int UADDR = 0xF6E;
public int POSTDEC0 = 0xFED;
public int T1CON = 0xFCD;
public int TXREG = 0xFAD;
public int LATE = 0xF8D;
public int UCON = 0xF6D;
public int PREINCO = 0xFEC;
public int TMR2 = 0xFCC;
public int TXSTA = 0xFAC;
public int LATD = 0xF8C;
public int USTAT = 0xF6C;
public int PLUSW = 0xFEB;
public int PR2 = 0xFCB;
public int RCSTA = 0xFAB;
public int LATC = 0xF8B;
public int UEIE = 0xF6B;
public int FSR0H = 0FEA;
public int T2CON = 0xFCA;
public int LATB = 0xF8A;
public int UEIR = 0xF6A;
public int FSR0L = 0FE9;
public int SSPBUF = 0FC9;
public int EEADR = 0FA9;
public int LATA = 0F89;
public int UIE = 0XF69;
public int WREG = 0FE8;
public int SSPADD = 0FC8;
public int EEDATA = 0FA8;
public int UIR = 0XF68;
public int INDF1 = 0FE7;
```

```

public int SSPSTAT = 0xFC7;
public int EECN2 = 0xFA7;
public int UFRMH = 0xF67;
public int POSTINC1 = 0xFE6;
public int SSPCON1 = 0xFC6;
public int EECN1 = 0xFA6;
public int UFRML = 0xF66;
public int POSTDEC1 = 0xFE5;
public int SSPCON2 = 0xFC5;
public int SPPCON = 0xF65;
public int PREINC1 = 0xFE4;
public int ADRESH = 0xFC4;
public int PORTE = 0xF84;
public int SPPEPS = 0xF64;
public int PLUSW1 = 0xFE3;
public int ADRESL = 0xFC3;
public int PORTD = 0xF83;
public int SPPCFG = 0xF63;
public int FSR1H = 0xFE2;
public int ADCON0 = 0xFC2;
public int IPR2 = 0xFA2;
public int PORTC = 0xF82;
public int SPPDATA = 0xF62;
public int FSR1L = 0xFE1;
public int ADCON1 = 0xFC1;
public int PIR2 = 0xFA1;
public int PORTB = 0xF81;
public int BSR = 0xFE0;
public int ADCON2 = 0xFC0;
public int PIE2 = 0xFA0;
public int PORTA = 0xF80;

public Pic18f2550Ram(List<MemoryRange> gpRegisters, List<MemoryRange>
sfRegisters) {
    super(gpRegisters, sfRegisters);
}

public static Pic18f2550Ram create() {
    Pic18f2550Ram ram;
    List<MemoryRange> sfrs, gprs;
    sfrs = new ArrayList<>();
    gprs = new ArrayList<>();

    gprs.add(new MemoryRange(0x60, 0xFF));
    gprs.add(new MemoryRange(0x100, 0x1FF));
    gprs.add(new MemoryRange(0x200, 0x2FF));
    gprs.add(new MemoryRange(0x300, 0x3FF));
    gprs.add(new MemoryRange(0x400, 0x4FF));
    gprs.add(new MemoryRange(0x500, 0x5FF));
    gprs.add(new MemoryRange(0x600, 0x6FF));
    gprs.add(new MemoryRange(0x700, 0x7FF));

    sfrs.add(new MemoryRange(0xF62, 0xF84));
    sfrs.add(new MemoryRange(0xF89, 0xF8D));
    sfrs.add(new MemoryRange(0xF92, 0xF96));
    sfrs.add(new MemoryRange(0xF9B, 0xF9B));
    sfrs.add(new MemoryRange(0xF9D, 0xFA2));
}

```

```

        sfrs.add(new MemoryRange(0xFA6, 0xFA9));
        sfrs.add(new MemoryRange(0xFAB, 0xFB8));
        sfrs.add(new MemoryRange(0xFBA, 0xFD3));
        sfrs.add(new MemoryRange(0xFD5, 0xFFFF));

        ram = new Pic18f2550Ram(gprs, sfrs);
        return ram;
    }

    public boolean containsSFR(int address) {
        boolean contains = false;
        for (int i = 0; i < sfRegisters.size(); i++) {
            contains = sfRegisters.get(i).contains(address);
        }
        return contains;
    }

    public boolean containsGPR(int address) {
        boolean contains = false;
        for (int i = 0; i < gpRegisters.size(); i++) {
            contains = gpRegisters.get(i).contains(address);
        }
        return contains;
    }
}

package com.vts.jplceditor.compiler.pic18.mem;

/**
 *
 * @author Vusivus
 */
public class ProgramMemoryModel {

    private final MemoryRange programMemory;
    private final MemoryRange idLocations;
    private final MemoryRange configFuses;
    private final MemoryRange deviceId;
    private final MemoryRange eepromData;

    public ProgramMemoryModel(MemoryRange programMemory, MemoryRange
idLocations, MemoryRange configFuses, MemoryRange deviceId, MemoryRange
eepromData) {
        this.programMemory = programMemory;
        this.idLocations = idLocations;
        this.configFuses = configFuses;
        this.deviceId = deviceId;
        this.eepromData = eepromData;
    }

    public static ProgramMemoryModel forPic18f2550() {
        ProgramMemoryModel memory;
        MemoryRange code = new MemoryRange(0x20, 0x7FFF);
        MemoryRange idlocs = new MemoryRange(0x200000, 0x200007);
        MemoryRange conf = new MemoryRange(0x300000, 0x30000d);
        MemoryRange id = new MemoryRange(0x3ffffe, 0x3fffff);
    }
}

```

```
    MemoryRange eeprom = new MemoryRange(0xf00000, 0xf000ff);
    memory = new ProgramMemoryModel(code, idlocs, conf, id, eeprom);
    return memory;
}

public MemoryRange getProgramMemory() {
    return programMemory;
}

public MemoryRange getIdLocations() {
    return idLocations;
}

public MemoryRange getConfigFuses() {
    return configFuses;
}

public MemoryRange getDeviceId() {
    return deviceId;
}

public MemoryRange getEepromData() {
    return eepromData;
}

}

package com.vts.jplceditor.compiler.pic18.mem;

import java.util.List;

/**
 *
 * @author Vusivus
 */
public class RamMemoryModel {

    final List<MemoryRange> gpRegisters;
    final List<MemoryRange> sfRegisters;

    public RamMemoryModel(List<MemoryRange> gpRegisters, List<MemoryRange>
sfRegisters) {
        this.gpRegisters = gpRegisters;
        this.sfRegisters = sfRegisters;
    }

    public List<MemoryRange> getGpRegisters() {
        return gpRegisters;
    }

    public List<MemoryRange> getSfRegisters() {
        return sfRegisters;
    }

}
package com.vts.jplceditor.compiler.plc;
```

```
/*
 *
 * @author Vusivus
 */
public class PLCOperatorDecoder {

    public static PLCOperator fromID(String instructionID) {
        PLCOperator instruction = null;
        switch (instructionID) {
            case "XIC":
                instruction = PLCOperator.XIC;
                break;
            case "XIO":
                instruction = PLCOperator.XIO;
                break;
            case "OTE":
                instruction = PLCOperator.OTE;
                break;
            case "OTL":
                instruction = PLCOperator.OTL;
                break;
            case "OTU":
                instruction = PLCOperator.OTU;
                break;
            case "TON":
                instruction = PLCOperator.TON;
                break;
            case "CTU":
                instruction = PLCOperator.CTU;
                break;
            case "CTD":
                instruction = PLCOperator.CTD;
                break;
            case "EQU":
                instruction = PLCOperator.EQU;
                break;
            case "GRT":
                instruction = PLCOperator.GRT;
                break;
            case "LES":
                instruction = PLCOperator.LES;
                break;
            case "LIM":
                instruction = PLCOperator.LIM;
                break;
            case "MOV":
                instruction = PLCOperator.MOV;
                break;
            case "MVM":
                instruction = PLCOperator.MVM;
                break;
            case "AND":
                instruction = PLCOperator.AND;
                break;
            case "NOT":
                instruction = PLCOperator.NOT;
        }
    }
}
```

```
        break;
    case "OR":
        instruction = PLCOperator.OR;
        break;
    case "XOR":
        instruction = PLCOperator.XOR;
        break;
    case "ADD":
        instruction = PLCOperator.ADD;
        break;
    case "DIV":
        instruction = PLCOperator.DIV;
        break;
    case "SUB":
        instruction = PLCOperator.SUB;
        break;
    case "MUL":
        instruction = PLCOperator.MUL;
        break;
    }
    return instruction;
}
}

package com.vts.jplceditor.compiler.plc;

/**
 *
 * @author Vusivus
 */
public enum PLCOperator {
    XIC,
    XIO,
    OTE,
    OTU,
    OTL,
    TON,
    CTU,
    EQU,
    LES,
    GRT,
    ADD,
    SUB,
    DIV,
    MUL,
    SQR,
    AND,
    NOT,
    OR,
    XOR,
    MOV,
    MVM,
    COP,
    JMP,
    LBL,
    SBR,
```

```
JSR,
RET,
SUS,
END,
NULL,
LIM,
CTD
}

package com.vts.jplceditor.compiler.plc;

import com.vts.jplceditor.compiler.pic18.PicAddress;

/**
 *
 * @author Vusivus
 */
public class PLCOperand {

    private PLCDATAType dataType;
    private String identity;
    private PicAddress address;

    public PLCOperand(PLCDATAType dataType, String identity) {
        this.dataType = dataType;
        this.identity = identity;
    }

    public PLCDATAType getDataType() {
        return dataType;
    }

    public void setDataType(PLCDATAType dataType) {
        this.dataType = dataType;
    }

    public String getIdentity() {
        return identity;
    }

    public void setIdentity(String identity) {
        this.identity = identity;
        if (isInteger(identity)) {
            dataType = PLCDATAType.LITERAL;
        }
    }

    public PicAddress getAddress() {
        return address;
    }

    public void setAddress(PicAddress address) {
        this.address = address;
    }

    public static boolean isInteger(String s) {
```

```
    if (s.isEmpty()) {
        return false;
    }
    for (int i = 0; i < s.length(); i++) {
        if (i == 0 && s.charAt(i) == '-') {
            if (s.length() == 1) {
                return false;
            } else {
                continue;
            }
        }
        if (Character.digit(s.charAt(i), 10) < 0) {
            return false;
        }
    }
    return true;
}
}
```

```
package com.vts.jplceditor.compiler.plc;

import com.vts.jplceditor.compiler.pic18.PicAddress;

/**
 *
 * @author Vusivus
 */
public class PLCOperand {

    private PLCDATAType dataType;
    private String identity;
    private PicAddress address;

    public PLCOperand(PLCDATAType dataType, String identity) {
        this.dataType = dataType;
        this.identity = identity;
    }

    public PLCDATAType getDataType() {
        return dataType;
    }

    public void setDataType(PLCDATAType dataType) {
        this.dataType = dataType;
    }

    public String getIdentity() {
        return identity;
    }

    public void setIdentity(String identity) {
        this.identity = identity;
        if (isInteger(identity)) {
            dataType = PLCDATAType.LITERAL;
```

```
        }

    }

    public PicAddress getAddress() {
        return address;
    }

    public void setAddress(PicAddress address) {
        this.address = address;
    }

    public static boolean isInteger(String s) {
        if (s.isEmpty()) {
            return false;
        }
        for (int i = 0; i < s.length(); i++) {
            if (i == 0 && s.charAt(i) == '-') {
                if (s.length() == 1) {
                    return false;
                } else {
                    continue;
                }
            }
            if (Character.digit(s.charAt(i), 10) < 0) {
                return false;
            }
        }
        return true;
    }
}
```

```
package com.vts.jplceditor.compiler.plc;

/**
 *
 * @author Vusivus
 */
public enum PLCDataType {
    BIT(0),
    BYTE(1),
    INT(2),
    FLOAT(4),
    LITERAL(2);

    private final int byteCount;

    private PLCDataType(int byteCount) {
        this.byteCount = byteCount;
    }

    public int getByteCount() {
        return byteCount;
    }
}
```

}

```

package com.vts.jplceditor.compiler.plc.instruction;

import com.vts.jplceditor.compiler.pic18.BitInstruction;
import com.vts.jplceditor.compiler.pic18.ByteInstruction;
import com.vts.jplceditor.compiler.pic18.ControlInstruction;
import com.vts.jplceditor.compiler.pic18.LiteralInstruction;
import com.vts.jplceditor.compiler.pic18.Opcode;
import com.vts.jplceditor.compiler.pic18.Pic18Instruction;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.plc.PLCDATAType;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusman
 */
public class PLC_Add extends Pic18Mem implements PLCInstr {

    private final List<Pic18Instruction> asm;
    private final int bitAdress;
    private int org;
    private final byte EN = 0;
    private final byte ENO = 1;
    private int wordCount;
    private int count;
    private final PLCOOperand prevENO, dest;
    private final int byteAndByteSize = 14;
    private final int byteAndIntSize = 14;
    private final int intAndIntSize = 14;

    public PLC_Add(int org, List<PLCOOperand> parameters, PLCOOperand prevENO)
    {

        this.bitAdress = parameters.get(0).getAddress().getFileAddr();
        asm = new ArrayList<>();
        this.org = org;
        PLCOOperand in1, in2;
        in1 = parameters.get(2);
        in2 = parameters.get(3);
        dest = parameters.get(4);
        this.prevENO = prevENO;

        if (in1.getDataType() == PLCDATAType.BYTE && in2.getDataType() ==
PLCDATAType.BYTE) {
            if (dest.getDataType() == PLCDATAType.INT) {
                checkENO(byteAndByteSize + 4);
            } else {
                checkENO(byteAndByteSize);
            }
            byteAndByte(in1, in2);
        }
    }
}

```

```

        } else if (in1.getDataType() == PLCDataType.BYTE && in2.getDataType()
== PLCDataType.INT) {
            if (dest.getDataType() == PLCDataType.INT) {
                checkENO(byteAndByteSize + 10);
            } else {
                checkENO(byteAndByteSize);
            }
            byteAndInt(in1, in2);
        } else if (in1.getDataType() == PLCDataType.INT && in2.getDataType()
== PLCDataType.BYTE) {
            if (dest.getDataType() == PLCDataType.INT) {
                checkENO(byteAndByteSize + 10);
            } else {
                checkENO(byteAndByteSize);
            }
            byteAndInt(in2, in1);
        } else if (in1.getDataType() == PLCDataType.INT && in2.getDataType()
== PLCDataType.INT) {
            if (dest.getDataType() == PLCDataType.INT) {
                checkENO(byteAndByteSize + 10);
            } else {
                checkENO(byteAndByteSize);
            }
            intAndInt(in1, in2);
        }
    }

private void checkENO(int endif) {
    int code_start, clearENO;

    //First set the EN depending on the prev ENO
    if (prevENO == null) { //If no prev ENO, this is rung start
        banksel(bitAdress);
        asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN, bsr));
        wordCount = 2;
    } else { // Else set this EN depending on the prev ENO
        code_start = org + 18;
        clearENO = code_start + endif; //clear_ENO in instruction code
        int clrEN = org + 12;
        int enFile = prevENO.getAddress().getFileAddr();
        /*0*/ banksel(enFile);
        /*2*/ asm.add(new BitInstruction(Opcode.BTFSS, enFile, ENO,
bsr)); // If ENO set, set EN
        /*4*/ branch(org + 4, clrEN);
        /*6*/ banksel(bitAdress);
        /*8*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN,
bsr));
        /*10*/ branch(org + 10, code_start);

        //clrEN address
        /*12*/ banksel(bitAdress);
        /*14*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, EN,
bsr));
        /*16*/ branch(org + 16, clearENO);
        wordCount = 9;
    }
}

```

```

        org += (wordCount * 2);
    }

private void byteAndByte(PLCOperand in1, PLCOperand in2) {
    int input1, input2, des;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();
    des = dest.getAddress().getFileAddr();

    //IF EN
    //DEST = IN1+IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*2*/ banksel(input1);
    /*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input1, bsr, w));
    /*6*/ banksel(input2);
    /*8*/ asm.add(new ByteInstruction(Opcode.ADDWF, input2, bsr, w));
    /*10*/ banksel(des);
    /*12*/ asm.add(new ByteInstruction(Opcode.MOVWF, des, bsr, false));
    wordCount = 7;
    if (dest.getDataType() == PLCDataType.INT) {
        /*14*/ asm.add(new LiteralInstruction(Opcode.MOVLW, 0));
        /*16*/ asm.add(new ByteInstruction(Opcode.ADDWFC, des + 1, bsr,
w));
    }
    wordCount += 2;
}

private void byteAndInt(PLCOperand in1, PLCOperand in2) {

    int input1, input2, des;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();
    des = dest.getAddress().getFileAddr();

    //IF EN
    //DEST = IN1+IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*2*/ banksel(input1);
    /*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input1, bsr, w));
    /*6*/ banksel(input2);
    /*8*/ asm.add(new ByteInstruction(Opcode.ADDWF, input2, bsr, w));
    /*10*/ banksel(des);
    /*12*/ asm.add(new ByteInstruction(Opcode.MOVWF, des, bsr, false));

    wordCount = 7;
    if (dest.getDataType() == PLCDataType.INT) {
        /*14*/ banksel(WREG);
        /*16*/ asm.add(new ByteInstruction(Opcode.CLRF, WREG, bsr,
false));
        /*18*/ banksel(input2);
        /*20*/ asm.add(new ByteInstruction(Opcode.ADDWFC, input2 + 1,
bsr, w));
    }
}

```

```

        /*22*/ asm.add(new ByteInstruction(Opcode.MOVWF, des + 1, bsr,
false));
        wordCount += 5;
    }
}

private void intAndInt(PLCOperand in1, PLCOperand in2) {
    int input1, input2, des;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();
    des = dest.getAddress().getFileAddr();

    //IF EN
    //DEST = IN1+IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*2*/ banksel(input1);
    /*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input1, bsr, w));
    /*6*/ banksel(input2);
    /*8*/ asm.add(new ByteInstruction(Opcode.ADDWF, input2, bsr, w));
    /*10*/ banksel(des);
    /*12*/ asm.add(new ByteInstruction(Opcode.MOVWF, des, bsr, false));
    wordCount = 7;
    if (dest.getDataType() == PLCDataType.INT) {
        /*14*/ banksel(input1);
        /*16*/ asm.add(new ByteInstruction(Opcode.MOVF, input1 + 1, bsr,
w));
        /*18*/ banksel(input2);
        /*20*/ asm.add(new ByteInstruction(Opcode.ADDWFC, input2 + 1,
bsr, w));
        /*22*/ asm.add(new ByteInstruction(Opcode.MOVWF, des + 1, bsr,
false));
        wordCount += 5;
    }
}

@Override
public List<Pic18Instruction> getPicAsm() {
    return asm;
}

@Override
public int getwordCount() {
    return wordCount;
}

private void banksel(int file) {
    byte bank = (byte) (file >> 8);
    asm.add(new LiteralInstruction(Opcode.MOVLB, bank));
}

private void branch(int currentAddress, int targetAddress) {
    //address = PC + 2 + 2n
    int branchAddress = targetAddress - currentAddress - 2;
    branchAddress /= 2;
    asm.add(new ControlInstruction(Opcode.BRA, branchAddress));
}

```

```

        }
    }

package com.vts.jplceditor.compiler.plc.instruction;

import com.vts.jplceditor.compiler.pic18.BitInstruction;
import com.vts.jplceditor.compiler.pic18.ByteInstruction;
import com.vts.jplceditor.compiler.pic18.ControlInstruction;
import com.vts.jplceditor.compiler.pic18.LiteralInstruction;
import com.vts.jplceditor.compiler.pic18Opcode;
import com.vts.jplceditor.compiler.pic18Pic18Instruction;
import com.vts.jplceditor.compiler.pic18Pic18Mem;
import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCOperand;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusman
 */
public class PLC_And extends Pic18Mem implements PLCInstr {
    private final List<Pic18Instruction> asm;
    private final int bitAdress;
    private int org;
    private final byte EN = 0;
    private final byte ENO = 1;
    private int wordCount;
    private final PLCOperand prevENO, dest;

    public PLC_And(int org, List<PLCOperand> parameters, PLCOperand prevENO)
    {

        this.bitAdress = parameters.get(0).getAddress().getFileAddr();
        asm = new ArrayList<>();
        this.org = org;
        PLCOperand in1, in2;
        in1 = parameters.get(2);
        in2 = parameters.get(3);
        dest = parameters.get(4);
        this.prevENO = prevENO;

        if (in1.getDataType() == PLCDatatype.BYTE && in2.getDataType() ==
        PLCDatatype.BYTE) {
            checkENO();
            byteAndByte(in1, in2);
        } else if (in1.getDataType() == PLCDatatype.BYTE && in2.getDataType() ==
        PLCDatatype.INT) {
            checkENO();
            byteAndInt(in1, in2);
        } else if (in1.getDataType() == PLCDatatype.INT && in2.getDataType() ==
        PLCDatatype.BYTE) {
            checkENO();
        }
    }
}

```

```

        byteAndInt(in2, in1);
    } else if (in1.getDataType() == PLCDataType.INT && in2.getDataType()
== PLCDataType.INT) {
        checkENO();
        intAndInt(in1, in2);
    }
}

private void checkENO() {
    int code_start;
    int clrEN = org+12;
    //First set the EN depending on the prev ENO
    if (prevENO == null) { //If no prev ENO, this is rung start
        banksel(bitAdress);
        asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN, bsr));
        wordCount = 2;
    } else { // Else set this EN depending on the prev ENO
        code_start = org+16;
        wordCount = 8;
        int enFile = prevENO.getAddress().getFileAddr();
        /*0*/ banksel(enFile);
        /*2*/ asm.add(new BitInstruction(Opcode.BTFSS, enFile, ENO,
bsr)); // If ENO set, set EN
        /*4*/ branch(org+4,clrEN);
        /*6*/ banksel(bitAdress);
        /*8*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN,
bsr));
        /*10*/ branch(org+10,code_start);

        //clrEN address
        /*12*/ banksel(bitAdress);
        /*14*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, EN,
bsr));
    }
    org += (wordCount*2);
}

private void byteAndByte(PLCOperand in1, PLCOperand in2) {
    int input1, input2, des;
    int endif = org+10;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();
    des = dest.getAddress().getFileAddr();

    //IF EN
    //DEST = IN1&IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BTFSS, bitAdress, EN, bsr));
// If EN set EN, execute code
    /*2*/ branch(org+2,endif);
    /*4*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*6*/ banksel(input1);
    /*8*/ asm.add(new ByteInstruction(Opcode.MOVF, input1, bsr, w));
    /*10*/ banksel(input2);
    /*12*/ asm.add(new ByteInstruction(Opcode.ANDWF, input2, bsr, w));
    /*14*/ banksel(des);
}

```

```

/*16*/ asm.add(new ByteInstruction(Opcode.MOVWF, des, bsr, false));
wordCount = 7;
if (dest.getDataType() == PLCDataType.INT) {
    /*18*/ asm.add(new LiteralInstruction(Opcode.MOVLW, 0));
    /*20*/ asm.add(new ByteInstruction(Opcode.ADDWFC, des + 1, bsr,
w));
    wordCount += 2;
}
wordCount = 10;
}

private void byteAndInt(PLCOperand in1, PLCOperand in2) {

    int input1, input2, des;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();
    des = dest.getAddress().getFileAddr();

    //IF EN
    //DEST = IN1&IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*2*/ banksel(input1);
    /*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input1, bsr, w));
    /*6*/ banksel(input2);
    /*8*/ asm.add(new ByteInstruction(Opcode.ANDWF, input2, bsr, w));
    /*10*/ banksel(des);
    /*12*/ asm.add(new ByteInstruction(Opcode.MOVWF, des, bsr, false));

    wordCount = 7;
    if (dest.getDataType() == PLCDataType.INT) {
        /*14*/ banksel(des+1);
        /*16*/ asm.add(new ByteInstruction(Opcode.CLRF, des+1, bsr,
false));
        wordCount += 4;
    }
    wordCount = 11;
}

private void intAndInt(PLCOperand in1, PLCOperand in2) {
    int input1, input2, des;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();
    des = dest.getAddress().getFileAddr();

    //IF EN
    //DEST = IN1&IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*1*/ banksel(input1);
    /*2*/ asm.add(new ByteInstruction(Opcode.MOVF, input1, bsr, w));
    /*3*/ banksel(input2);
    /*4*/ asm.add(new ByteInstruction(Opcode.ANDWF, input2, bsr, w));
    /*5*/ banksel(des);
    /*6*/ asm.add(new ByteInstruction(Opcode.MOVWF, des, bsr, false));
}

```

```

        wordCount = 7;
        if (dest.getDataType() == PLCDataType.INT) {
            /*7*/ banksel(input1);
            /*8*/ asm.add(new ByteInstruction(Opcode.MOVF, input1+1, bsr,
w));
            /*9*/ banksel(input2);
            /*10*/ asm.add(new ByteInstruction(Opcode.ANDWF, input2 + 1, bsr,
w));
            /*11*/ banksel(des);
            /*12*/ asm.add(new ByteInstruction(Opcode.MOVWF, des+1, bsr,
false));
            wordCount += 6;
        }
        /*17*/ asm.add(new ControlInstruction(Opcode.NOP));      //endif
address
        wordCount = 17;
    }

@Override
public List<Pic18Instruction> getPicAsm() {
    return asm;
}

@Override
public int getwordCount() {
    return wordCount;
}

private void banksel(int file) {
    byte bank = (byte) (file >> 8);
    asm.add(new LiteralInstruction(Opcode.MOVLB, bank));
}

private void branch(int currentAddress, int targetAddress) {
    //address = PC + 2 + 2n
    int branchAddress = targetAddress - currentAddress - 2;
    branchAddress /= 2;
    asm.add(new ControlInstruction(Opcode.BRA, branchAddress));
}
}

package com.vts.jplceditor.compiler.plc.instruction;

import com.vts.jplceditor.compiler.pic18.BitInstruction;
import com.vts.jplceditor.compiler.pic18.ByteInstruction;
import com.vts.jplceditor.compiler.pic18.ControlInstruction;
import com.vts.jplceditor.compiler.pic18.LiteralInstruction;
import com.vts.jplceditor.compiler.pic18.Opcode;
import com.vts.jplceditor.compiler.pic18.Pic18Instruction;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCOperand;
import java.util.ArrayList;

```

```

import java.util.List;

/**
 *
 * @author Vusman
 */
public class PLC_Equ extends Pic18Mem implements PLCInstr {

    private final List<Pic18Instruction> asm;
    private final int bitAdress;
    private int org, count;
    private final byte EN = 0;
    private final byte ENO = 1;
    private int wordCount;
    private final PLCOOperand prevENO;

    public PLC_Equ(int org, List<PLCOOperand> parameters, PLCOOperand prevENO)
    {

        this.bitAdress = parameters.get(0).getAddress().getFileAddr();
        asm = new ArrayList<>();
        this.org = org;
        PLCOOperand in1, in2;
        in1 = parameters.get(2);
        in2 = parameters.get(3);
        this.prevENO = prevENO;

        if (in1.getDataType() == PLCDATA_TYPE.BYTE && in2.getDataType() ==
        PLCDATA_TYPE.BYTE) {
            checkENO();
            byteToByte(in1, in2);
        } else if (in1.getDataType() == PLCDATA_TYPE.BYTE && in2.getDataType() ==
        PLCDATA_TYPE.INT) {
            checkENO();
            byteToInt(in1, in2);
        } else if (in1.getDataType() == PLCDATA_TYPE.INT && in2.getDataType() ==
        PLCDATA_TYPE.BYTE) {
            checkENO();
            intToByte(in1, in2);
        } else if (in1.getDataType() == PLCDATA_TYPE.INT && in2.getDataType() ==
        PLCDATA_TYPE.INT) {
            checkENO();
            intToInt(in1, in2);
        }
    }

    private void checkENO() {
        int code_start;
        int clrEN = (org + 8)*2;
        //First set the EN depending on the prev ENO
        if (prevENO == null) { //If no prev ENO, this is rung start
            banksel(bitAdress);
            asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN, bsr));
            count = 2;
        } else { // Else set this EN depending on the prev ENO
            code_start = (9 + org)*2;
            count = 9;
        }
    }
}

```

```

        int enFile = prevENO.getAddress().getFileAddr();
        /*0*/ banksel(enFile);
        /*1*/ asm.add(new BitInstruction(Opcode.BTFSS, enFile, ENO,
bsr)); // If ENO set, set EN
        /*2*/ asm.add(new ControlInstruction(Opcode.GOTO, clrEN));
        /*4*/ banksel(bitAdress);
        /*5*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN,
bsr));
        /*6*/ asm.add(new ControlInstruction(Opcode.GOTO, code_start));
        //clrEN address
        /*8*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, EN,
bsr));
    }
    org += (count*2);
}

private void byteToByte(PLCOperand in1, PLCOperand in2) {
    int endif = (15 + org)*2;
    int not_equal = (org + 14)*2;
    int input1, input2;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();

    //IF EN
    //ENO = IN1==IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BTFSS, bitAdress, EN, bsr));
    /*1*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));      //
Goto ENDIF if EN=0
    /*3*/ banksel(input1);
    /*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input1, bsr, w));
    /*5*/ banksel(input2);
    /*6*/ asm.add(new ByteInstruction(Opcode.SUBWF, input2, bsr, w));
    /*7*/ banksel(STATUS);
    /*8*/ asm.add(new BitInstruction(Opcode.BTFSC, STATUS, Z, bsr));
//if in2=in1 Z=1;
    /*9*/ asm.add(new ControlInstruction(Opcode.GOTO, not_equal));
    /*11*/ banksel(bitAdress);
    /*12*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*13*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));

    //not_equal address
    /*14*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, ENO, bsr));
    /*15*/ asm.add(new ControlInstruction(Opcode.NOP));      //endif
address
    wordCount = 16+count;
}

private void byteToInt(PLCOperand in1, PLCOperand in2) {

    int equal = (org + 18)*2;
    int not_equal = (org + 22)*2;
    int endif = (not_equal + 2)*2;
    int input1, input2;
    input1 = in1.getAddress().getFileAddr();

```

```

        input2 = in2.getAddress().getFileAddr();

        //IF EN
        //ENO = IN1==IN2
        //ENDIF
        /*0*/ asm.add(new BitInstruction(Opcode.BTFSS, bitAdress, EN, bsr));
        /*1*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));
// Goto ENDIF if EN=0
        /*3*/ banksel(input2);
        /*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input2 + 1, bsr, w));
//check if in2H=0
        /*5*/ asm.add(new LiteralInstruction(Opcode.SUBLW, 0));
        /*6*/ banksel(STATUS);
        /*7*/ asm.add(new BitInstruction(Opcode.BTFSC, STATUS, Z, bsr));
//Check if 0-in2H produces negative
        /*8*/ asm.add(new ControlInstruction(Opcode.GOTO, not_equal));

        //now check lower byte of int with than of byte
        /*10*/ banksel(input2);
        /*11*/ asm.add(new ByteInstruction(Opcode.MOVF, input2, bsr, w));
        /*12*/ banksel(input1);
        /*13*/ asm.add(new ByteInstruction(Opcode.SUBWF, input1, bsr, w));
//in1-in2L
        /*14*/ banksel(STATUS);
        /*15*/ asm.add(new BitInstruction(Opcode.BTFSC, STATUS, Z, bsr));
//if zero, equal
        /*16*/ asm.add(new ControlInstruction(Opcode.GOTO, not_equal));

        //here is the equal address
        /*18*/ banksel(bitAdress);
        /*19*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO,
bsr)); //if in1<in2 set ENO
        /*20*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));

        //here is the not_equal address
        /*22*/ banksel(bitAdress);
        /*23*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, ENO,
bsr)); //if greater than zero set ENO
        /*24*/ asm.add(new ControlInstruction(Opcode.NOP)); //endif address
wordCount = 25+count;
}

private void intToByte(PLCOperand in1, PLCOperand in2) {
    int equal = (org + 18)*2;
    int not_equal = (org + 22)*2;
    int endif = (not_equal + 2)*2;
    int input1, input2;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();

    //IF EN
    //ENO = IN1==IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BTFSS, bitAdress, EN, bsr));

```

```

        /*1*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));
// Goto ENDIF if EN=0
        /*3*/ banksel(input1);
        /*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input1 + 1, bsr, w));
//check if in1H>0
        /*5*/ asm.add(new LiteralInstruction(Opcode.SUBLW, 0));
        /*6*/ banksel(STATUS);
        /*7*/ asm.add(new BitInstruction(Opcode.BTFSC, STATUS, Z, bsr));
//Check if 0-in1H produces zero
        /*8*/ asm.add(new ControlInstruction(Opcode.GOTO, not_equal));

        //now check lower byte of int with than of byte
        /*10*/ banksel(input2);
        /*11*/ asm.add(new ByteInstruction(Opcode.MOVF, input2, bsr, w));
        /*12*/ banksel(input1);
        /*13*/ asm.add(new ByteInstruction(Opcode.SUBWF, input1, bsr, w));
//in1-in2L
        /*14*/ banksel(STATUS);
        /*15*/ asm.add(new BitInstruction(Opcode.BTFSC, STATUS, Z, bsr));
//if zero, equal
        /*16*/ asm.add(new ControlInstruction(Opcode.GOTO, not_equal));

        //here is the equal address
        /*18*/ banksel(bitAdress);
        /*19*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO,
a));//if in1<in2 set ENO
        /*20*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));

        //here is the not_equal address
        /*22*/ banksel(bitAdress);
        /*23*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, ENO,
a));//if greater than zero set ENO
        /*24*/ asm.add(new ControlInstruction(Opcode.NOP));    //endif address
wordCount = 25+count;
}

private void intToInt(PLCOperand in1, PLCOperand in2) {
    int equal = (org + 19)*2;
    int not_equal = (org + 23)*2;
    int endif = (not_equal + 2)*2;
    int input1, input2;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();

    //IF EN
    //ENO = IN1==IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BTFSS, bitAdress, EN, bsr));
    /*1*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));
// Goto ENDIF if EN=0
    /*3*/ banksel(input2);
    /*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input2 + 1, bsr, w));
//check if in1H>in2H
    /*5*/ banksel(input1);

```

```

        /*6*/ asm.add(new ByteInstruction(Opcode.SUBWF, input1 + 1, bsr, w));
//in1H-in2H
        /*7*/ banksel(STATUS);
        /*8*/ asm.add(new BitInstruction(Opcode.BTFSC, STATUS, Z, bsr));
//Check if in1H-in2H produces zero
        /*9*/ asm.add(new ControlInstruction(Opcode.GOTO, not_equal));

        //now check lower bytes
        /*11*/ banksel(input2);
        /*12*/ asm.add(new ByteInstruction(Opcode.MOVF, input2, bsr, w));
        /*13*/ banksel(input1);
        /*14*/ asm.add(new ByteInstruction(Opcode.SUBWF, input1, bsr, w));
//in1L-in2L
        /*15*/ banksel(STATUS);
        /*16*/ asm.add(new BitInstruction(Opcode.BTFSS, STATUS, Z, bsr));
//if zero, equal
        /*17*/ asm.add(new ControlInstruction(Opcode.GOTO, not_equal));

        //here is the equal address
        /*19*/ banksel(bitAdress);
        /*20*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO,
bsr)); //if in1<in2 set ENO
        /*21*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));

        //here is the not_equal address
        /*23*/ banksel(bitAdress);
        /*24*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, ENO,
bsr)); //if greater than zero set ENO
        /*25*/ asm.add(new ControlInstruction(Opcode.NOP)); //endif address
wordCount = 26+count;
}

@Override
public List<Pic18Instruction> getPicAsm() {
    return asm;
}

@Override
public int getwordCount() {
    return wordCount;
}

private void banksel(int file) {
    byte bank = (byte) (file >> 8);
    asm.add(new LiteralInstruction(Opcode.MOVLB, bank));
}
}

```

```

package com.vts.jplceditor.compiler.plc.instruction;

import com.vts.jplceditor.compiler.pic18.BitInstruction;
import com.vts.jplceditor.compiler.pic18.ByteInstruction;
import com.vts.jplceditor.compiler.pic18.ControlInstruction;

```

```

import com.vts.jplceditor.compiler.pic18.LiteralInstruction;
import com.vts.jplceditor.compiler.pic18.Opcode;
import com.vts.jplceditor.compiler.pic18.Pic18Instruction;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCOperand;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusman
 */
public class PLC_Grt extends Pic18Mem implements PLCIstr {

    private final List<Pic18Instruction> asm;
    private final int bitAdress;
    int org;
    private final byte EN = 0;
    private final byte ENO = 1;
    private int wordCount;
    private int count;
    private final PLCOperand prevENO;

    public PLC_Grt(int org, List<PLCOperand> parameters, PLCOperand prevENO)
    {

        this.bitAdress = parameters.get(0).getAddress().getFileAddr();
        asm = new ArrayList<>();
        this.org = org;
        PLCOperand in1, in2;
        in1 = parameters.get(2);
        in2 = parameters.get(3);
        this.prevENO = prevENO;

        if (in1.getDataType() == PLCDatatype.BYTE && in2.getDataType() ==
PLCDatatype.BYTE) {
            checkENO();
            byteToByte(in1, in2);
        } else if (in1.getDataType() == PLCDatatype.BYTE && in2.getDataType() ==
PLCDatatype.INT) {
            checkENO();
            byteToInt(in1, in2);
        } else if (in1.getDataType() == PLCDatatype.INT && in2.getDataType() ==
PLCDatatype.BYTE) {
            checkENO();
            intToByte(in1, in2);
        } else if (in1.getDataType() == PLCDatatype.INT && in2.getDataType() ==
PLCDatatype.INT) {
            checkENO();
            intToInt(in1, in2);
        }
    }

    private void checkENO() {
        int code_start;
        int clrEN = (org + 8)*2;
    }
}

```

```

//First set the EN depending on the prev ENO
if (prevENO == null) { //If no prev ENO, this is rung start
    banksel(bitAdress);
    asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN, bsr));
    count = 2;
} else { // Else set this EN depending on the prev ENO
    code_start = (9 + org)*2;
    count = 9;
    int enFile = prevENO.getAddress().getFileAddr();
    /*0*/ banksel(enFile);
    /*1*/ asm.add(new BitInstruction(Opcode.BTFSS, enFile, ENO,
bsr)); // If ENO set, set EN
    /*2*/ asm.add(new ControlInstruction(Opcode.GOTO, clrEN));
    /*4*/ banksel(bitAdress);
    /*5*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN,
bsr));
    /*6*/ asm.add(new ControlInstruction(Opcode.GOTO, code_start));

    //clrEN address
    /*8*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, EN,
bsr));
}
org += (count*2);
}

private void byteToByte(PLCOperand in1, PLCOperand in2) {
    int endif = (org + 17)*2;
    int not_greater = (org + 18)*2;
    int input1, input2;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();

    //IF EN
    //ENO = IN1>IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BTFSS, bitAdress, EN, bsr));
    /*1*/ asm.add(new ControlInstruction(Opcode.GOTO, endif)); // Goto ENDIF if EN=0
    /*3*/ banksel(input1);
    /*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input1, bsr, w));
    /*5*/ banksel(input2);
    /*6*/ asm.add(new ByteInstruction(Opcode.SUBWF, input2, bsr, w));
    /*7*/ banksel(STATUS);
    /*8*/ asm.add(new BitInstruction(Opcode.BTFSC, STATUS, C, bsr));
//if in2<in1 C=1;
    /*9*/ asm.add(new ControlInstruction(Opcode.GOTO, not_greater));
    /*11*/ banksel(bitAdress);
    /*12*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*13*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));

    //not_greater address
    /*15*/ banksel(bitAdress);
    /*16*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, ENO, bsr));
    /*17*/ asm.add(new ControlInstruction(Opcode.NOP)); //endif
address
wordCount = 18+count;
}

```

```

private void byteToInt(PLCOperand in1, PLCOperand in2) {

    int greater = (org + 18)*2;
    int not_greater = (org + 22)*2;
    int endif = (not_greater + 2)*2;
    int input1, input2;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();

    //IF EN
    //ENO = IN1>IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BTFSS, bitAdress, EN, bsr));
    /*1*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));

// Goto ENDIF if EN=0
/*3*/ banksel(input2);
/*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input2 + 1, bsr, w));
//check if in2H>0
/*5*/ asm.add(new LiteralInstruction(Opcode.SUBLW, 0));
/*6*/ banksel(STATUS);
/*7*/ asm.add(new BitInstruction(Opcode.BTFSS, STATUS, N, bsr));
//Check if 0-in2H produces negative
/*8*/ asm.add(new ControlInstruction(Opcode.GOTO, not_greater));

//now check lower byte of int with than of byte
/*10*/ banksel(input2);
/*11*/ asm.add(new ByteInstruction(Opcode.MOVF, input2, a, bsr));
/*12*/ banksel(input1);
/*13*/ asm.add(new ByteInstruction(Opcode.SUBWF, input1, a, bsr));
//in1-in2L
/*14*/ banksel(STATUS);
/*15*/ asm.add(new BitInstruction(Opcode.BTFSC, STATUS, N, bsr));
//if negative, not greater
/*16*/ asm.add(new ControlInstruction(Opcode.GOTO, not_greater));

//here is the greater address
/*18*/ banksel(bitAdress);
/*19*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO,
bsr)); //if in1<in2 set ENO
/*20*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));

//here is the not_greater address
/*22*/ banksel(bitAdress);
/*23*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, ENO,
bsr)); //if greater than zero set ENO
/*24*/ asm.add(new ControlInstruction(Opcode.NOP)); //endif address
wordCount = 25+count;
}

private void intToByte(PLCOperand in1, PLCOperand in2) {
    int greater = (org + 18)*2;
    int not_greater = (org + 22)*2;
    int endif = (not_greater + 2)*2;
}

```

```

        int input1,input2;
        input1 = in1.getAddress().getFileAddr();
        input2 = in2.getAddress().getFileAddr();

        //IF EN
        //ENO = IN1>IN2
        //ENDIF
        /*0*/ asm.add(new BitInstruction(Opcode.BTFSS, bitAdress, EN, a));
        /*1*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));
// Goto ENDIF if EN=0
/*3*/ banksel(input1);
/*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input1 + 1, a, w));
//check if in1H>0
/*5*/ asm.add(new LiteralInstruction(Opcode.SUBLW, 0));
/*6*/ banksel(STATUS);
/*7*/ asm.add(new BitInstruction(Opcode.BTFSC, STATUS, N, a));
//Check if 0-in1H produces negative
/*8*/ asm.add(new ControlInstruction(Opcode.GOTO, greater));

        //now check lower byte of int with than of byte
/*10*/ banksel(input2);
/*11*/ asm.add(new ByteInstruction(Opcode.MOVF, input2, a, w));
/*12*/ banksel(input1);
/*13*/ asm.add(new ByteInstruction(Opcode.SUBWF, input1, a, w));
//in1-in2L
/*14*/ banksel(STATUS);
/*15*/ asm.add(new BitInstruction(Opcode.BTFSC, STATUS, N, a));
//if negative, not greater
/*16*/ asm.add(new ControlInstruction(Opcode.GOTO, not_greater));

        //here is the greater address
/*18*/ banksel(bitAdress);
/*19*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO,
a)); //if in1<in2 set ENO
/*20*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));

        //here is the not_greater address
/*22*/ banksel(bitAdress);
/*23*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, ENO,
a)); //if greater than zero set ENO
/*24*/ asm.add(new ControlInstruction(Opcode.NOP)); //endif address
wordCount = 25+count;
}

private void intToInt(PLCOperand in1, PLCOperand in2) {
    int greater = (org + 19)*2;
    int not_greater = (org + 23)*2;
    int endif = (not_greater + 2)*2;
    int input1,input2;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();

    //IF EN
    //ENO = IN1>IN2

```

```

        //ENDIF
        /*0*/ asm.add(new BitInstruction(Opcode.BTFSS, bitAdress, EN, bsr));
        /*1*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));
// Goto ENDIF if EN=0
        /*3*/ banksel(input2);
        /*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input2 + 1, bsr, w));
//check if in1H>in2H
        /*5*/ banksel(input1);
        /*6*/ asm.add(new ByteInstruction(Opcode.SUBWF, input1+1, bsr, w));
//in1H-in2H
        /*7*/ banksel(STATUS);
        /*8*/ asm.add(new BitInstruction(Opcode.BTFSC, STATUS, N, bsr));
//Check if in1H-in2H produces negative
        /*9*/ asm.add(new ControlInstruction(Opcode.GOTO, greater));

        //now check lower bytes
        /*11*/ banksel(input2);
        /*12*/ asm.add(new ByteInstruction(Opcode.MOVF, input2, bsr, w));
        /*13*/ banksel(input1);
        /*14*/ asm.add(new ByteInstruction(Opcode.SUBWF, input1, bsr, w));
//in1L-in2L
        /*15*/ banksel(STATUS);
        /*16*/ asm.add(new BitInstruction(Opcode.BTFSS, STATUS, N, bsr));
//if negative, not less
        /*17*/ asm.add(new ControlInstruction(Opcode.GOTO, not_greater));

        //here is the greater address
        /*19*/ banksel(bitAdress);
        /*20*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO,
bsr)); //if in1<in2 set ENO
        /*21*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));

        //here is the not_greater address
        /*23*/ banksel(bitAdress);
        /*24*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, ENO,
bsr)); //if greater than zero set ENO
        /*25*/ asm.add(new ControlInstruction(Opcode.NOP)); //endif address
wordCount = 26+count;
}

@Override
public List<Pic18Instruction> getPicAsm() {
    return asm;
}

@Override
public int getwordCount() {
    return wordCount;
}

private void banksel(int file){
    byte bank = (byte) (file>>8);
    asm.add(new LiteralInstruction(Opcode.MOVLB,bank));
}

```

}

```

package com.vts.jplceditor.compiler.plc.instruction;

import com.vts.jplceditor.compiler.pic18.BitInstruction;
import com.vts.jplceditor.compiler.pic18.ByteInstruction;
import com.vts.jplceditor.compiler.pic18.ControlInstruction;
import com.vts.jplceditor.compiler.pic18.LiteralInstruction;
import com.vts.jplceditor.compiler.pic18.Opcode;
import com.vts.jplceditor.compiler.pic18.Pic18Instruction;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusman
 */
public class PLC_Les extends Pic18Mem implements PLCInstr {

    private final List<Pic18Instruction> asm;
    private final int bitAdress;
    private int org;
    private final byte EN = 0;
    private final byte ENO = 1;
    private int wordCount;
    private int count;
    private final PLCOOperand prevENO;

    public PLC_Les(int org, List<PLCOOperand> parameters, PLCOOperand prevENO)
    {

        this.bitAdress = parameters.get(0).getAddress().getFileAddr();
        asm = new ArrayList<>();
        this.org = org;
        PLCOOperand in1, in2;
        in1 = parameters.get(2);
        in2 = parameters.get(3);
        this.prevENO = prevENO;

        if (in1.getDataType() == PLCDatatype.BYTE && in2.getDataType() ==
PLCDatatype.BYTE) {
            checkENO();
            byteToByte(in1, in2);
        } else if (in1.getDataType() == PLCDatatype.BYTE && in2.getDataType() ==
PLCDatatype.INT) {
            checkENO();
            byteToInt(in1, in2);
        } else if (in1.getDataType() == PLCDatatype.INT && in2.getDataType() ==
PLCDatatype.BYTE) {
            checkENO();
            intToByte(in1, in2);
        } else if (in1.getDataType() == PLCDatatype.INT && in2.getDataType() ==
PLCDatatype.INT) {
    }
}

```

```

        checkENO();
        intToInt(in1, in2);
    }

}

private void checkENO() {
    int code_start;
    int clrEN = (org + 8)*2;
    //First set the EN depending on the prev ENO
    if (prevENO == null) { //If no prev ENO, this is rung start
        banksel(bitAdress);
        asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN, bsr));
        count = 2;
    } else { // Else set this EN depending on the prev ENO
        code_start = (9 + org)*2;
        count = 9;
        int enFile = prevENO.getAddress().getFileAddr();
        /*0*/ banksel(enFile);
        /*1*/ asm.add(new BitInstruction(Opcode.BTFSS, enFile, ENO,
bsr)); // If ENO set, set EN
        /*2*/ asm.add(new ControlInstruction(Opcode.GOTO, clrEN));
        /*4*/ banksel(bitAdress);
        /*5*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN,
bsr));
        /*6*/ asm.add(new ControlInstruction(Opcode.GOTO, code_start));

        //clrEN address
        /*8*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, EN,
bsr));
    }
    org += (count*2);
}

private void byteToByte(PLCOperand in1, PLCOperand in2) {
    int endif = (org + 17)*2;
    int not_less = (org + 15)*2;
    int input1, input2;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();

    //IF EN
    //ENO = IN1<IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BTFSS, bitAdress, EN, bsr));
    /*1*/ asm.add(new ControlInstruction(Opcode.GOTO, endif)); // Goto ENDIF if EN=0
    /*3*/ banksel(input1);
    /*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input1, bsr, w));
    /*5*/ banksel(input2);
    /*6*/ asm.add(new ByteInstruction(Opcode.SUBWF, input2, bsr, w));
    /*7*/ banksel(STATUS);
    /*8*/ asm.add(new BitInstruction(Opcode.BTFSS, STATUS, C, bsr));
//if in2<in1 C=1;
    /*9*/ asm.add(new ControlInstruction(Opcode.GOTO, not_less));
    /*11*/ banksel(bitAdress);
    /*12*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*13*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));
}

```

```

        //not_less address
        /*15*/ banksel(bitAdress);
        /*16*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, ENO, bsr));
        /*17*/ asm.add(new ControlInstruction(Opcode.NOP));      //endif
address
    wordCount = (18+count);
}

private void byteToInt(PLCOperand in1, PLCOperand in2) {

    int less = (org + 18)*2;
    int not_less = (org + 22)*2;
    int endif = (not_less + 2)*2;
    int input1, input2;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();

    //IF EN
    //ENO = IN1<IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BTFSS, bitAdress, EN, bsr));
    /*1*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));
// Goto ENDIF if EN=0
    /*3*/ banksel(input2);
    /*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input2 + 1, bsr, w));
//check if in2H>0
    /*5*/ asm.add(new LiteralInstruction(Opcode.SUBLW, 0));
    /*6*/ banksel(STATUS);
    /*7*/ asm.add(new BitInstruction(Opcode.BTFSC, STATUS, N, bsr));
//Check if 0-in2H produces negative
    /*8*/ asm.add(new ControlInstruction(Opcode.GOTO, less));

    //now check lower byte of int with than of byte
    /*10*/ banksel(input2);
    /*11*/ asm.add(new ByteInstruction(Opcode.MOVF, input2, bsr, w));
    /*12*/ banksel(input1);
    /*13*/ asm.add(new ByteInstruction(Opcode.SUBWF, input1, bsr, w));
//in1-in2L
    /*14*/ banksel(STATUS);
    /*15*/ asm.add(new BitInstruction(Opcode.BTFSC, STATUS, N, bsr));
//if negative,in1 less than in2
    /*16*/ asm.add(new ControlInstruction(Opcode.GOTO, not_less));

    //here is the less address
    /*18*/ banksel(bitAdress);
    /*19*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO,
bsr)); //if in1<in2 set ENO
    /*20*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));

    //here is the not_less address
    /*22*/ banksel(bitAdress);
    /*23*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, ENO,
bsr)); //if greater than zero set ENO
}

```

```

/*24*/ asm.add(new ControlInstruction(Opcode.NOP)); //endif address
wordCount = (25+count);
}

private void intToByte(PLCOperand in1, PLCOperand in2) {
    int less = (org + 18)*2;
    int not_less = (org + 22)*2;
    int endif = (not_less + 2)*2;
    int input1, input2;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();

    //IF EN
    //ENO = IN1<IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BTFSS, bitAdress, EN, bsr));
    /*1*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));
// Goto ENDIF if EN=0
    /*3*/ banksel(input1);
    /*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input1 + 1, bsr, w));
//check if in1H>0
    /*5*/ asm.add(new LiteralInstruction(Opcode.SUBLW, 0));
    /*6*/ banksel(STATUS);
    /*7*/ asm.add(new BitInstruction(Opcode.BTFSC, STATUS, N, bsr));
//Check if 0-in1H produces negative
    /*8*/ asm.add(new ControlInstruction(Opcode.GOTO, not_less));

    //now check lower byte of int with than of byte
    /*10*/ banksel(input2);
    /*11*/ asm.add(new ByteInstruction(Opcode.MOVF, input2, bsr, w));
    /*12*/ banksel(input1);
    /*13*/ asm.add(new ByteInstruction(Opcode.SUBWF, input1, bsr, w));
//in1-in2L
    /*14*/ banksel(STATUS);
    /*15*/ asm.add(new BitInstruction(Opcode.BTFSC, STATUS, N, bsr));
//if negative, not less
    /*16*/ asm.add(new ControlInstruction(Opcode.GOTO, not_less));

    //here is the less address
    /*18*/ banksel(bitAdress);
    /*19*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO,
bsr)); //if in1<in2 set ENO
    /*20*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));

    //here is the not_less address
    /*22*/ banksel(bitAdress);
    /*23*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, ENO,
bsr)); //if greater than zero set ENO
    /*24*/ asm.add(new ControlInstruction(Opcode.NOP)); //endif address
wordCount = (25+count);
}

private void intToInt(PLCOperand in1, PLCOperand in2) {
    int less = (org + 19)*2;

```

```

        int not_less = (org + 23)*2;
        int endif = (not_less + 2)*2;
        int input1, input2;
        input1 = in1.getAddress().getFileAddr();
        input2 = in2.getAddress().getFileAddr();

        //IF EN
        //ENO = IN1<IN2
        //ENDIF
        /*0*/ asm.add(new BitInstruction(Opcode.BTFSS, bitAdress, EN, a));
        /*1*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));
// Goto ENDIF if EN=0
/*3*/ banksel(input2);
/*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input2 + 1, a, w));
//check if in1H>in2H
/*5*/ banksel(input1);
/*6*/ asm.add(new ByteInstruction(Opcode.SUBWF, input1+1, a, w));
//in1H-in2H
/*7*/ banksel(STATUS);
/*8*/ asm.add(new BitInstruction(Opcode.BTFSC, STATUS, N, a));
//Check if in1H-in2H produces negative
/*9*/ asm.add(new ControlInstruction(Opcode.GOTO, not_less));

        //now check lower bytes
/*11*/ banksel(input2);
/*12*/ asm.add(new ByteInstruction(Opcode.MOVF, input2, a, w));
/*13*/ banksel(input1);
/*14*/ asm.add(new ByteInstruction(Opcode.SUBWF, input1, a, w));
//in1L-in2L
/*15*/ banksel(STATUS);
/*16*/ asm.add(new BitInstruction(Opcode.BTFSC, STATUS, N, a));
//if negative, not less
/*17*/ asm.add(new ControlInstruction(Opcode.GOTO, not_less));

        //here is the less address
/*19*/ banksel(bitAdress);
/*20*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO,
a)); //if in1<in2 set ENO
/*21*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));

        //here is the not_less address
/*23*/ banksel(bitAdress);
/*24*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, ENO,
a)); //if greater than zero set ENO
/*25*/ asm.add(new ControlInstruction(Opcode.NOP)); //endif address
wordCount = 26+count;
}

@Override
public List<Pic18Instruction> getPicAsm() {
    return asm;
}

@Override

```

```

public int getwordCount() {
    return wordCount;
}

private void banksel(int file){
    byte bank = (byte) (file>>8);
    asm.add(new LiteralInstruction(Opcode.MOVLB, bank));
}
}

package com.vts.jplceditor.compiler.plc.instruction;

import com.vts.jplceditor.compiler.pic18.BitInstruction;
import com.vts.jplceditor.compiler.pic18.ByteInstruction;
import com.vts.jplceditor.compiler.pic18.ControlInstruction;
import com.vts.jplceditor.compiler.pic18.LiteralInstruction;
import com.vts.jplceditor.compiler.pic18.Opcode;
import com.vts.jplceditor.compiler.pic18.Pic18Instruction;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.plc.PLCDATAType;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusman
 */
public class Plc_Mov extends Pic18Mem implements PLCInstr {

    private final List<Pic18Instruction> asm;
    private final int bitAdress;
    private int org, count;
    private final byte EN = 0;
    private final byte ENO = 1;
    private int wordCount;
    private final PLCOOperand prevENO;

    public Plc_Mov(int org, List<PLCOOperand> parameters, PLCOOperand prevENO)
    {

        this.bitAdress = parameters.get(0).getAddress().getFileAddr();
        asm = new ArrayList<>();
        this.org = org;
        PLCOOperand in1, in2;
        in1 = parameters.get(2);
        in2 = parameters.get(3);
        this.prevENO = prevENO;

        if (in1.getDataType() == PLCDATAType.BYTE) {
            checkENO();
            moveByte(in1, in2);
        } else if (in1.getDataType() == PLCDATAType.INT) {
            checkENO();
        }
    }
}

```

```

        moveInt(in1, in2);
    } else if (in1.getDataType() == PLCDATAType.INT) {
        checkENO();
        moveLiteral(in1, in2);
    }
}

private void checkENO() {
    int code_start;
    int clrEN = (org + 8)*2;
    //First set the EN depending on the prev ENO
    if (prevENO == null) { //If no prev ENO, this is rung start
        banksel(bitAdress);
        asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN, bsr));
        count = 2;
    } else { // Else set this EN depending on the prev ENO
        code_start = (9 + org)*2;
        count = 9;
        int enFile = prevENO.getAddress().getFileAddr();
        /*0*/ banksel(enFile);
        /*1*/ asm.add(new BitInstruction(Opcode.BTFSS, enFile, ENO,
bsr)); // If ENO set, set EN
        /*2*/ asm.add(new ControlInstruction(Opcode.GOTO, clrEN));
        /*4*/ banksel(bitAdress);
        /*5*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN,
bsr));
        /*6*/ asm.add(new ControlInstruction(Opcode.GOTO, code_start));

        //clrEN address
        /*8*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, EN,
bsr));
    }
    org += (count*2);
}

private void moveByte(PLCOperand in1, PLCOperand in2) {
    int endif = (8 + org)*2;
    int source, dest;
    source = in1.getAddress().getFileAddr();
    dest = in2.getAddress().getFileAddr();

    //ENO = TRUE
    //IF EN
    //DEST = SOURCE
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*1*/ asm.add(new BitInstruction(Opcode.BTFSS, bitAdress, EN, bsr));
    /*2*/ asm.add(new ControlInstruction(Opcode.GOTO, endif)); // Goto ENDIF if EN=0
    /*4*/ banksel(source);
    /*5*/ asm.add(new ByteInstruction(Opcode.MOVF, source, bsr, w));
    /*6*/ banksel(dest);
    /*7*/ asm.add(new ByteInstruction(Opcode.MOVWF, dest, bsr, false));
    /*8*/ asm.add(new ControlInstruction(Opcode.NOP)); //endif address
    wordCount = 9 + count;
}

```

```

private void moveInt(PLCOperand in1, PLCOperand in2) {
    int endif = (org + 7)*2;
    int source, dest;
    source = in1.getAddress().getFileAddr();
    dest = in2.getAddress().getFileAddr();
    if (in2.getDataType() == PLCDATA_TYPE.INT) {
        endif += 5;
    }

    //ENO = TRUE
    //IF EN
    //DEST = SOURCE
    //ENDIF
    /*0*/ asm.add(new BitInstruction(OpCode.BSF, bitAdress, ENO, bsr));
    /*1*/ asm.add(new BitInstruction(OpCode.BTFSS, bitAdress, EN, bsr));
    /*2*/ asm.add(new ControlInstruction(OpCode.GOTO, endif));      //
Goto ENDIF if EN=0
    /*4*/ banksel(source);
    /*5*/ asm.add(new ByteInstruction(OpCode.MOVF, source, bsr, w));
    /*6*/ banksel(dest);
    /*7*/ asm.add(new ByteInstruction(OpCode.MOVWF, dest, bsr, false));
    if (in2.getDataType() == PLCDATA_TYPE.INT) {
        /*8*/ banksel(source + 1);
        /*9*/ asm.add(new ByteInstruction(OpCode.MOVF, source + 1, bsr,
w));
        /*10*/ banksel(dest + 1);
        /*11*/ asm.add(new ByteInstruction(OpCode.MOVWF, dest + 1, bsr,
false));
        /*12*/ asm.add(new ControlInstruction(OpCode.NOP));      //endif
address
        wordCount += 5;
    }
    wordCount = 8 + count;
}

private void moveLiteral(PLCOperand in1, PLCOperand in2) {
    int endif = (org + 6)*2;
    int source, dest;
    source = Integer.valueOf(in1.getIdentity());
    dest = in2.getAddress().getFileAddr();
    if (in2.getDataType() == PLCDATA_TYPE.INT) {
        endif += 4;
    }

    //ENO = TRUE
    //IF EN
    //DEST = SOURCE
    //ENDIF
    /*0*/ asm.add(new BitInstruction(OpCode.BSF, bitAdress, ENO, bsr));
    /*1*/ asm.add(new BitInstruction(OpCode.BTFSS, bitAdress, EN, bsr));
    /*2*/ asm.add(new ControlInstruction(OpCode.GOTO, endif));      //
Goto ENDIF if EN=0
    /*4*/ asm.add(new LiteralInstruction(OpCode.MOVLW, 0xFF & source));
    /*5*/ banksel(dest);
}

```

```

        /*6*/ asm.add(new ByteInstruction(Opcode.MOVWF, dest, bsr, false));
        if (in2.getDataType() == PLCDATAType.INT) {
            /*7*/ asm.add(new LiteralInstruction(Opcode.MOVLW, (source &
0xFF00) >> 8));
            /*8*/ banksel(dest + 1);
            /*9*/ asm.add(new ByteInstruction(Opcode.MOVWF, dest + 1, bsr,
false));
            /*10*/ asm.add(new ControlInstruction(Opcode.NOP)); //endif
address
        wordCount += 4;
    }
    wordCount = 7 + count;
}

@Override
public List<Pic18Instruction> getPicAsm() {
    return asm;
}

@Override
public int getwordCount() {
    return wordCount;
}

private void banksel(int file) {
    byte bank = (byte) (file >> 8);
    asm.add(new LiteralInstruction(Opcode.MOVLB, bank));
}
}

```

```

package com.vts.jplceditor.compiler.plc.instruction;

import com.vts.jplceditor.compiler.pic18.BitInstruction;
import com.vts.jplceditor.compiler.pic18.ByteInstruction;
import com.vts.jplceditor.compiler.pic18.ControlInstruction;
import com.vts.jplceditor.compiler.pic18.LiteralInstruction;
import com.vts.jplceditor.compiler.pic18.Opcode;
import com.vts.jplceditor.compiler.pic18.Pic18Instruction;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.plc.PLCDATAType;
import com.vts.jplceditor.compiler.plc.PLCOperand;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusman
 */
public class Plc_Mvm extends Pic18Mem implements PLCInstr {

    private final List<Pic18Instruction> asm;
    private final int bitAdress;
    private int org, count;

```

```

private final byte EN = 0;
private final byte ENO = 1;
private int wordCount;
private final PLCOOperand prevENO;

public Plc_Mvm(int org, List<PLCOOperand> parameters, PLCOOperand prevENO)
{

    this.bitAdress = parameters.get(0).getAddress().getFileAddr();
    asm = new ArrayList<>();
    this.org = org;
    PLCOOperand in1, in2, in3;
    in1 = parameters.get(2);
    in2 = parameters.get(3);
    in3 = parameters.get(4);
    this.prevENO = prevENO;

    if (in1.getDataType() == PLCDatatype.BYTE) {
        checkENO();
        moveByte(in1, in2, in3);
    } else if (in1.getDataType() == PLCDatatype.INT) {
        checkENO();
        moveInt(in1, in2, in3);
    } else if (in1.getDataType() == PLCDatatype.INT) {
        checkENO();
        moveLiteral(in1, in2, in3);
    }
}

private void checkENO() {
    int code_start;
    int clrEN = (org + 8)*2;
    //First set the EN depending on the prev ENO
    if (prevENO == null) { //If no prev ENO, this is rung start
        banksel(bitAdress);
        asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN, bsr));
        count = 2;
    } else { // Else set this EN depending on the prev ENO
        code_start = (9 + org)*2;
        count = 9;
        int enFile = prevENO.getAddress().getFileAddr();
        /*0*/ banksel(enFile);
        /*1*/ asm.add(new BitInstruction(Opcode.BTFSS, enFile, ENO,
        bsr)); // If ENO set, set EN
        /*2*/ asm.add(new ControlInstruction(Opcode.GOTO, clrEN));
        /*4*/ banksel(bitAdress);
        /*5*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN,
        bsr));
        /*6*/ asm.add(new ControlInstruction(Opcode.GOTO, code_start));

        //clrEN address
        /*8*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, EN,
        bsr));
    }
    org += (count*2);
}

```

```

private void moveByte(PLCOperand in1, PLCOperand in2, PLCOperand in3) {
    int endif = (9 + org)*2;
    int source, mask, dest;
    source = in1.getAddress().getFileAddr();
    mask = Integer.valueOf(in2.getIdentity());
    dest = in3.getAddress().getFileAddr();

    //ENO = TRUE
    //IF EN
    //DEST = SOURCE
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*1*/ asm.add(new BitInstruction(Opcode.BTFSS, bitAdress, EN, bsr));
    /*2*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));      //
Goto ENDIF if EN=0
    /*4*/ banksel(source);
    /*5*/ asm.add(new ByteInstruction(Opcode.MOVF, source, bsr, w));
    /*6*/ asm.add(new LiteralInstruction(Opcode.ANDLW, mask));
    /*7*/ banksel(dest);
    /*8*/ asm.add(new ByteInstruction(Opcode.MOVWF, dest, bsr, false));
    /*9*/ asm.add(new ControlInstruction(Opcode.NOP));      //endif address
    wordCount = 10 + count;
}

private void moveInt(PLCOperand in1, PLCOperand in2, PLCOperand in3) {
    int endif = (org + 8)*2;
    int source, mask, dest;
    source = in1.getAddress().getFileAddr();
    mask = Integer.valueOf(in2.getIdentity());
    dest = in3.getAddress().getFileAddr();
    if (in2.getDataType() == PLCDATA_TYPE.INT) {
        endif += 6;
    }

    //ENO = TRUE
    //IF EN
    //DEST = SOURCE
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*1*/ asm.add(new BitInstruction(Opcode.BTFSS, bitAdress, EN, bsr));
    /*2*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));      //
Goto ENDIF if EN=0
    /*4*/ banksel(source);
    /*5*/ asm.add(new ByteInstruction(Opcode.MOVF, source, bsr, w));
    /*6*/ asm.add(new LiteralInstruction(Opcode.ANDLW, mask));
    /*7*/ banksel(dest);
    /*8*/ asm.add(new ByteInstruction(Opcode.MOVWF, dest, bsr, false));
    if (in3.getDataType() == PLCDATA_TYPE.INT) {
        /*9*/ banksel(source + 1);
        /*10*/ asm.add(new ByteInstruction(Opcode.MOVF, source + 1, bsr,
w));
        /*11*/ asm.add(new LiteralInstruction(Opcode.ANDLW, mask));
        /*12*/ banksel(dest);
        /*13*/ asm.add(new ByteInstruction(Opcode.MOVWF, dest, bsr,
false));
    }
}

```

```

        /*12*/ asm.add(new ControlInstruction(Opcode.NOP));      //endif
address
        wordCount += 6;
    }
    wordCount = 9 + count;
}

private void moveLiteral(PLCOperand in1, PLCOperand in2, PLCOperand in3)
{
    int endif = (org + 7)*2;
    int source, mask, dest;
    source = in1.getAddress().getFileAddr();
    mask = Integer.valueOf(in2.getIdentity());
    dest = in3.getAddress().getFileAddr();
    if (in2.getDataType() == PLCDATA_TYPE.INT) {
        endif += 5;
    }

    //ENO = TRUE
    //IF EN
    //DEST = SOURCE
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAddress, ENO, bsr));
    /*1*/ asm.add(new BitInstruction(Opcode.BTFSS, bitAddress, EN, bsr));
    /*2*/ asm.add(new ControlInstruction(Opcode.GOTO, endif));      //
Goto ENDIF if EN=0
    /*4*/ asm.add(new LiteralInstruction(Opcode.MOVlw, source & 0xFF));
    /*5*/ asm.add(new LiteralInstruction(Opcode.ANDlw, mask));
    /*6*/ banksel(dest);
    /*7*/ asm.add(new ByteInstruction(Opcode.MOVWF, dest, bsr, false));
    if (in3.getDataType() == PLCDATA_TYPE.INT) {
        /*8*/ asm.add(new LiteralInstruction(Opcode.MOVlw, (source &
0xFF00) >> 8));
        /*9*/ asm.add(new LiteralInstruction(Opcode.ANDlw, mask));
        /*10*/ banksel(dest);
        /*11*/ asm.add(new ByteInstruction(Opcode.MOVWF, dest, bsr,
false));
        /*12*/ asm.add(new ControlInstruction(Opcode.NOP));      //endif
address
        wordCount += 5;
    }
    wordCount = 8 + count;
}

@Override
public List<Pic18Instruction> getPicAsm() {
    return asm;
}

@Override
public int getwordCount() {
    return wordCount;
}

private void banksel(int file) {
    byte bank = (byte) (file >> 8);
}

```

```

        asm.add(new LiteralInstruction(Opcode.MOVLB, bank));
    }

}

package com.vts.jplceditor.compiler.plc.instruction;

import com.vts.jplceditor.compiler.pic18.BitInstruction;
import com.vts.jplceditor.compiler.pic18.ByteInstruction;
import com.vts.jplceditor.compiler.pic18.ControlInstruction;
import com.vts.jplceditor.compiler.pic18.LiteralInstruction;
import com.vts.jplceditor.compiler.pic18.Opcode;
import com.vts.jplceditor.compiler.pic18.Pic18Instruction;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusman
 */
public class PLC_Or extends Pic18Mem implements PLCInstr {

    private final List<Pic18Instruction> asm;
    private final int bitAdress;
    private int org;
    private final byte EN = 0;
    private final byte ENO = 1;
    private int wordCount;
    private int count;
    private final PLCOOperand prevENO, dest;

    public PLC_Or(int org, List<PLCOOperand> parameters, PLCOOperand prevENO)
    {

        this.bitAdress = parameters.get(0).getAddress().getFileAddr();
        asm = new ArrayList<>();
        this.org = org;
        PLCOOperand in1, in2;
        in1 = parameters.get(2);
        in2 = parameters.get(3);
        dest = parameters.get(4);
        this.prevENO = prevENO;

        if (in1.getDataType() == PLCDatatype.BYTE && in2.getDataType() ==
        PLCDatatype.BYTE) {
            checkENO();
            byteAndByte(in1, in2);
        } else if (in1.getDataType() == PLCDatatype.BYTE && in2.getDataType() ==
        PLCDatatype.INT) {
            checkENO();
            byteAndInt(in1, in2);
        }
    }
}

```

```

        } else if (in1.getDataType() == PLCDatatype.INT && in2.getDataType()
== PLCDatatype.BYTE) {
            checkENO();
            byteAndInt(in2, in1);
        } else if (in1.getDataType() == PLCDatatype.INT && in2.getDataType()
== PLCDatatype.INT) {
            checkENO();
            intAndInt(in1, in2);
        }
    }

private void checkENO() {
    int code_start;
    int clrEN = (org + 8)*2;
    //First set the EN depending on the prev ENO
    if (prevENO == null) { //If no prev ENO, this is rung start
        banksel(bitAdress);
        asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN, bsr));
        count = 2;
    } else { // Else set this EN depending on the prev ENO
        code_start = (9 + org)*2;
        count = 9;
        int enFile = prevENO.getAddress().getFileAddr();
        /*0*/ banksel(enFile);
        /*1*/ asm.add(new BitInstruction(Opcode.BTFSS, enFile, ENO,
bsr)); // If ENO set, set EN
        /*2*/ asm.add(new ControlInstruction(Opcode.GOTO, clrEN));
        /*4*/ banksel(bitAdress);
        /*5*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN,
bsr));
        /*6*/ asm.add(new ControlInstruction(Opcode.GOTO, code_start));

        //clrEN address
        /*8*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, EN,
bsr));
    }
    org += (count*2);
}

private void byteAndByte(PLCOperand in1, PLCOperand in2) {
    int input1, input2, des;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();
    des = dest.getAddress().getFileAddr();

    //IF EN
    //DEST = IN1&IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*1*/ banksel(input1);
    /*2*/ asm.add(new ByteInstruction(Opcode.MOVF, input1, bsr, w));
    /*3*/ banksel(input2);
    /*4*/ asm.add(new ByteInstruction(Opcode.IORWF, input2, bsr, w));
    /*5*/ banksel(des);
    /*6*/ asm.add(new ByteInstruction(Opcode.MOVWF, des, bsr, false));
    wordCount = 7;
}

```

```

    if (dest.getDataType() == PLCDataType.INT) {
        /*7*/ asm.add(new LiteralInstruction(Opcode.MOVlw, 0));
        /*8*/ asm.add(new ByteInstruction(Opcode.ADDWFC, des + 1, bsr,
w));
        wordCount += 2;
    }
    /*17*/ asm.add(new ControlInstruction(Opcode.NOP)); //endif
address
wordCount = 1 + count;
}

private void byteAndInt(PLCOperand in1, PLCOperand in2) {

    int input1, input2, des;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();
    des = dest.getAddress().getFileAddr();

    //IF EN
    //DEST = IN1&IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*1*/ banksel(input1);
    /*2*/ asm.add(new ByteInstruction(Opcode.MOVF, input1, bsr, w));
    /*3*/ banksel(input2);
    /*4*/ asm.add(new ByteInstruction(Opcode.IORWF, input2, bsr, w));
    /*5*/ banksel(des);
    /*6*/ asm.add(new ByteInstruction(Opcode.MOVWF, des, bsr, false));

    wordCount = 7;
    if (dest.getDataType() == PLCDataType.INT) {
        /*8*/ banksel(input2+1);
        /*9*/ asm.add(new ByteInstruction(Opcode.MOVF, input2 + 1, bsr,
w));
        /*10*/ banksel(des+1);
        /*11*/ asm.add(new ByteInstruction(Opcode.MOVWF, des+1, bsr,
false));
        wordCount += 4;
    }
    /*17*/ asm.add(new ControlInstruction(Opcode.NOP)); //endif
address
wordCount = 1 + count;
}

private void intAndInt(PLCOperand in1, PLCOperand in2) {
    int input1, input2, des;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();
    des = dest.getAddress().getFileAddr();

    //IF EN
    //DEST = IN1&IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*1*/ banksel(input1);
}

```

```

/*2*/ asm.add(new ByteInstruction(Opcode.MOVF, input1, bsr, w));
/*3*/ banksel(input2);
/*4*/ asm.add(new ByteInstruction(Opcode.IORWF, input2, bsr, w));
/*5*/ banksel(des);
/*6*/ asm.add(new ByteInstruction(Opcode.MOVWF, des, bsr, false));

wordCount = 7;
if (dest.getDataType() == PLCDataType.INT) {
    /*7*/ banksel(input1);
    /*8*/ asm.add(new ByteInstruction(Opcode.MOVF, input1+1, bsr,
w));
    /*9*/ banksel(input2);
    /*10*/ asm.add(new ByteInstruction(Opcode.IORWF, input2 + 1, bsr,
w));
    /*11*/ banksel(des);
    /*12*/ asm.add(new ByteInstruction(Opcode.MOVWF, des+1, bsr,
false));
    wordCount += 6;
}
/*17*/ asm.add(new ControlInstruction(Opcode.NOP));      //endif
address
wordCount = 1 + count;
}

@Override
public List<Pic18Instruction> getPicAsm() {
    return asm;
}

@Override
public int getwordCount() {
    return wordCount;
}

private void banksel(int file) {
    byte bank = (byte) (file >> 8);
    asm.add(new LiteralInstruction(Opcode.MOVLB, bank));
}
}

```

```
package com.vts.jplceditor.compiler.plc.instruction;

import com.vts.jplceditor.compiler.pic18.BitInstruction;
import com.vts.jplceditor.compiler.pic18.ControlInstruction;
import com.vts.jplceditor.compiler.pic18.LiteralInstruction;
import com.vts.jplceditor.compiler.pic18.Opcode;
import com.vts.jplceditor.compiler.pic18.Pic18Instruction;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import java.util.ArrayList;
import java.util.List;

/**
```

```

* @author Vusman
*/
public class PLC_Ote extends Pic18Mem implements PLCInstr {

    private final List<Pic18Instruction> asm;
    private final int bitAdress;
    private int org;
    private final byte EN = 0;
    private final byte ENO = 1;
    private int wordCount;
    private int count;
    private final PLCOOperand prevENO;

    public PLC_Ote(int org, List<PLCOOperand> parameters, PLCOOperand prevENO)
    {

        this.bitAdress = parameters.get(0).getAddress().getFileAddr();
        asm = new ArrayList<>();
        this.org = org;
        this.prevENO = prevENO;
        PLCOOperand in = parameters.get(2);
        checkENO();
        ote(in);
    }

    private void checkENO() {
        int code_start, clearENO;

        //First set the EN depending on the prev ENO
        if (prevENO == null) { //If no prev ENO, this is rung start
            banksel(bitAdress);
            asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN, bsr));
            wordCount = 2;
        } else { // Else set this EN depending on the prev ENO
            code_start = org + 18;
            clearENO = code_start + 8; //clear_ENO in instruction code
            int clrEN = org + 12;
            int enFile = prevENO.getAddress().getFileAddr();
            /*0*/ banksel(enFile);
            /*2*/ asm.add(new BitInstruction(Opcode.BTFSS, enFile, ENO,
        bsr)); // If ENO set, set EN
            /*4*/ branch(org + 4, clrEN);
            /*6*/ banksel(bitAdress);
            /*8*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN,
        bsr));
            /*10*/ branch(org + 10, code_start);

            //clrEN address
            /*12*/ banksel(bitAdress);
            /*14*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, EN,
        bsr));
            /*16*/ branch(org + 16, clearENO);
            wordCount = 9;
        }
        org += (wordCount * 2);
    }
}

```

```

private void ote(PLCOperand in) {

    int endif;
    int clear_ENO;
    byte bit = in.getAddress().getBitNo();
    int input = in.getAddress().getFileAddr();

    clear_ENO = org + 8;
    endif = org + 16;

    //IF EN
    //ENO = TRUE
    //BIT = TRUE
    //ELSE
    //BIT = FALSE
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*2*/ banksel(input);
    /*4*/ asm.add(new BitInstruction(Opcode.BSF, input, bit, bsr));
    wordCount+=3;
    if (prevENO != null) {
        /*6*/ branch(org + 6, endif);
        //clear ENO address
        /*8*/ banksel(input);
        /*10*/ asm.add(new BitInstruction(Opcode.BCF, input, bit, bsr));
        /*12*/ banksel(bitAdress);
        /*14*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, ENO,
bsr));
        wordCount += 5;
    }
}

@Override
public List<Pic18Instruction> getPicAsm() {
    return asm;
}

@Override
public int getwordCount() {
    return wordCount;
}

private void banksel(int file) {
    byte bank = (byte) (file >> 8);
    asm.add(new LiteralInstruction(Opcode.MOVLB, bank));
}

private void branch(int currentAddress, int targetAddress) {
    //address = PC + 2 + 2n
    int branchAddress = targetAddress - currentAddress - 2;
    branchAddress /= 2;
    asm.add(new ControlInstruction(Opcode.BRA, branchAddress));
}
}

```

```

package com.vts.jplceditor.compiler.plc.instruction;

import com.vts.jplceditor.compiler.pic18.BitInstruction;
import com.vts.jplceditor.compiler.pic18.ControlInstruction;
import com.vts.jplceditor.compiler.pic18.LiteralInstruction;
import com.vts.jplceditor.compiler.pic18.Opcode;
import com.vts.jplceditor.compiler.pic18.Pic18Instruction;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusman
 */
public class PLC_Otl extends Pic18Mem implements PLCInstr {
    private final List<Pic18Instruction> asm;
    private final int bitAdress;
    private int org;
    private final byte EN = 0;
    private final byte ENO = 1;
    private int wordCount;
    private int count;
    private final PLCOOperand prevENO;

    public PLC_Otl(int org, List<PLCOOperand> parameters, PLCOOperand prevENO)
    {

        this.bitAdress = parameters.get(0).getAddress().getFileAddr();
        asm = new ArrayList<>();
        this.org = org;
        this.prevENO = prevENO;
        PLCOOperand in = parameters.get(2);
        checkENO();
        otl(in);
    }

    private void checkENO() {
        int code_start, clearENO;

        //First set the EN depending on the prev ENO
        if (prevENO == null) { //If no prev ENO, this is rung start
            banksel(bitAdress);
            asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN, bsr));
            wordCount = 2;
        } else { // Else set this EN depending on the prev ENO
            code_start = org + 18;
            clearENO = code_start + 8; //clear_ENO in instruction code
            int clrEN = org + 12;
            int enFile = prevENO.getAddress().getFileAddr();
            /*0*/ banksel(enFile);
            /*2*/ asm.add(new BitInstruction(Opcode.BTFSS, enFile, ENO,
            bsr)); // If ENO set, set EN
            /*4*/ branch(org + 4, clrEN);
            /*6*/ banksel(bitAdress);
        }
    }
}

```

```

        /*8*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN,
bsr));
        /*10*/ branch(org + 10, code_start);

        //clrEN address
        /*12*/ banksel(bitAdress);
        /*14*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, EN,
bsr));
        /*16*/ branch(org + 16, clearENO);
        wordCount = 9;
    }
    org += (wordCount * 2);
}

private void otl(PLCOperand in) {
    int code_start;

    int endif;
    byte bit = in.getAddress().getBitNo();
    int input = in.getAddress().getFileAddr();

    //IF EN
    //ENO = TRUE
    //BIT = FALSE
    //ENDIF
    /*0*/ banksel(bitAdress);
    /*2*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*4*/ banksel(input);
    /*6*/ asm.add(new BitInstruction(Opcode.BSF, input, bit, bsr));
    wordCount += 4;
    if (prevENO != null) {
        //clearENO address
        /*8*/ banksel(bitAdress);
        /*10*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, ENO,
bsr));
        wordCount+=2;
    }
}

@Override
public List<Pic18Instruction> getPicAsm() {
    return asm;
}

@Override
public int getwordCount() {
    return wordCount;
}

private void banksel(int file) {
    byte bank = (byte) (file >> 8);
    asm.add(new LiteralInstruction(Opcode.MOVLB, bank));
}

```

```

private void branch(int currentAddress, int targetAddress) {
    //address = PC + 2 + 2n
    int branchAddress = targetAddress - currentAddress - 2;
    branchAddress /= 2;
    asm.add(new ControlInstruction(Opcode.BRA, branchAddress));
}
}

package com.vts.jplceditor.compiler.plc.instruction;

import com.vts.jplceditor.compiler.pic18.BitInstruction;
import com.vts.jplceditor.compiler.pic18.ControlInstruction;
import com.vts.jplceditor.compiler.pic18.LiteralInstruction;
import com.vts.jplceditor.compiler.pic18.Opcode;
import com.vts.jplceditor.compiler.pic18.Pic18Instruction;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusman
 */
public class PLC_Otu extends Pic18Mem implements PLCInstr {

    private final List<Pic18Instruction> asm;
    private final int bitAdress;
    private int org;
    private final byte EN = 0;
    private final byte ENO = 1;
    private int wordCount;
    private int count;
    private final PLCOOperand prevENO;

    public PLC_Otu(int org, List<PLCOOperand> parameters, PLCOOperand prevENO)
    {

        this.bitAdress = parameters.get(0).getAddress().getFileAddr();
        asm = new ArrayList<>();
        this.org = org;
        this.prevENO = prevENO;
        PLCOOperand in = parameters.get(2);
        checkENO();
        otu(in);
    }

    private void checkENO() {
        int code_start, clearENO;

        //First set the EN depending on the prev ENO
        if (prevENO == null) { //If no prev ENO, this is rung start
            banksel(bitAdress);
            asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN, bsr));
        }
    }
}

```

```

        wordCount = 2;
    } else { // Else set this EN depending on the prev ENO
        code_start = org + 18;
        clearENO = code_start + 8; //clear_ENO in instruction code
        int clrEN = org + 12;
        int enFile = prevENO.getAddress().getFileAddr();
        /*0*/ banksel(enFile);
        /*2*/ asm.add(new BitInstruction(Opcode.BTFSS, enFile, ENO,
bsr)); // If ENO set, set EN
        /*4*/ branch(org + 4, clrEN);
        /*6*/ banksel(bitAdress);
        /*8*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN,
bsr));
        /*10*/ branch(org + 10, code_start);

        //clrEN address
        /*12*/ banksel(bitAdress);
        /*14*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, EN,
bsr));
        /*16*/ branch(org + 16, clearENO);
        wordCount = 9;
    }
    org += (wordCount * 2);
}

private void otu(PLCOperand in) {
    byte bit = in.getAddress().getBitNo();
    int input = in.getAddress().getFileAddr();

    //IF EN
    //ENO = TRUE
    //BIT = FALSE
    //ENDIF
    /*0*/ banksel(bitAdress);
    /*2*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*4*/ banksel(input);
    /*6*/ asm.add(new BitInstruction(Opcode.BCF, input, bit, bsr));
    wordCount += 4;
    if (prevENO != null) {

        //clearENO address
        /*8*/ banksel(bitAdress);
        /*10*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, ENO,
bsr));
        wordCount += 2;
    }
}

@Override
public List<Pic18Instruction> getPicAsm() {
    return asm;
}

@Override
public int getwordCount() {

```

```
        return wordCount;
    }

    private void banksel(int file) {
        byte bank = (byte) (file >> 8);
        asm.add(new LiteralInstruction(Opcode.MOVLB, bank));
    }

    private void branch(int currentAddress, int targetAddress) {
        //address = PC + 2 + 2n
        int branchAddress = targetAddress - currentAddress - 2;
        branchAddress /= 2;
        asm.add(new ControlInstruction(Opcode.BRA, branchAddress));
    }
}

package com.vts.jplceditor.compiler.plc.instruction;

import com.vts.jplceditor.compiler.plc.PLCInstruction;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusman
 */
public class PLC_Rung {

    private List<PLCInstruction> instructions;

    public PLC_Rung() {
        instructions = new ArrayList<>();
    }

    public void addInstruction(PLCInstruction instruction) {
        instructions.add(instruction);
    }

    public List<PLCInstruction> getInstructions() {
        return instructions;
    }

    public void setInstructions(List<PLCInstruction> instructions) {
        this.instructions = instructions;
    }
}

package com.vts.jplceditor.compiler.plc.instruction;

import com.vts.jplceditor.compiler.pic18.BitInstruction;
import com.vts.jplceditor.compiler.pic18.ByteInstruction;
import com.vts.jplceditor.compiler.pic18.ControlInstruction;
import com.vts.jplceditor.compiler.pic18.LiteralInstruction;
import com.vts.jplceditor.compiler.pic18.Opcode;
```

```

import com.vts.jplceditor.compiler.pic18.Pic18Instruction;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusman
 */
public class PLC_Sub extends Pic18Mem implements PLCInstr {

    private final List<Pic18Instruction> asm;
    private final int bitAdress;
    private int org;
    private final byte EN = 0;
    private final byte ENO = 1;
    private int wordCount;
    private int count;
    private final PLCOOperand prevENO, dest;
    private final int byteAndByteSize = 14;
    private final int byteAndIntSize = 14;
    private final int intAndIntSize = 14;

    public PLC_Sub(int org, List<PLCOOperand> parameters, PLCOOperand prevENO)
    {

        this.bitAdress = parameters.get(0).getAddress().getFileAddr();
        asm = new ArrayList<>();
        this.org = org;
        PLCOOperand in1, in2;
        in1 = parameters.get(2);
        in2 = parameters.get(3);
        dest = parameters.get(4);
        this.prevENO = prevENO;

        if (in1.getDataType() == PLCDatatype.BYTE && in2.getDataType() ==
        PLCDatatype.BYTE) {
            if (dest.getDataType() == PLCDatatype.INT) {
                checkENO(byteAndByteSize + 2);
            } else {
                checkENO(byteAndByteSize);
            }
            byteAndByte(in1, in2);
        } else if (in1.getDataType() == PLCDatatype.BYTE && in2.getDataType() ==
        PLCDatatype.INT) {
            if (dest.getDataType() == PLCDatatype.INT) {
                checkENO(byteAndByteSize + 6);
            } else {
                checkENO(byteAndByteSize);
            }
            byteAndInt(in1, in2);
        } else if (in1.getDataType() == PLCDatatype.INT && in2.getDataType() ==
        PLCDatatype.BYTE) {
            if (dest.getDataType() == PLCDatatype.INT) {
                checkENO(byteAndByteSize + 6);
            }
        }
    }
}

```

```

        } else {
            checkENO(byteAndByteSize);
        }
        byteAndInt(in2, in1);
    } else if (in1.getDataType() == PLCDataType.INT && in2.getDataType() == PLCDataType.INT) {
        if (dest.getDataType() == PLCDataType.INT) {
            checkENO(byteAndByteSize + 10);
        } else {
            checkENO(byteAndByteSize);
        }
        intAndInt(in1, in2);
    }
}

private void checkENO(int endif) {
    int code_start, clearENO;

    //First set the EN depending on the prev ENO
    if (prevENO == null) { //If no prev ENO, this is rung start
        banksel(bitAdress);
        asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN, bsr));
        wordCount = 2;
    } else { // Else set this EN depending on the prev ENO
        code_start = org + 18;
        clearENO = code_start + endif; //clear_ENO in instruction code
        int clrEN = org + 12;
        int enFile = prevENO.getAddress().getFileAddr();
        /*0*/ banksel(enFile);
        /*2*/ asm.add(new BitInstruction(Opcode.BTFSS, enFile, ENO,
bsr)); // If ENO set, set EN
        /*4*/ branch(org + 4, clrEN);
        /*6*/ banksel(bitAdress);
        /*8*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN,
bsr));
        /*10*/ branch(org + 10, code_start);

        //clrEN address
        /*12*/ banksel(bitAdress);
        /*14*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, EN,
bsr));
        /*16*/ branch(org + 16, clearENO);
        wordCount = 9;
    }
    org += (wordCount * 2);
}

private void byteAndByte(PLCOperand in1, PLCOperand in2) {
    int input1, input2, des;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();
    des = dest.getAddress().getFileAddr();

    //IF EN
    //DEST = IN1+IN2
}

```

```

//ENDIF
/*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
/*2*/ banksel(input2);
/*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input2, bsr, w));
/*6*/ banksel(input1);
/*8*/ asm.add(new ByteInstruction(Opcode.SUBWF, input1, bsr, w));
/*10*/ banksel(des);
/*12*/ asm.add(new ByteInstruction(Opcode.MOVWF, des, bsr, false));
wordCount = 7;
if (dest.getDataType() == PLCDataType.INT) {
    /*14*/ asm.add(new ByteInstruction(Opcode.CLRF, des + 1, bsr,
false));
    wordCount += 1;
}
}

private void byteAndInt(PLCOperand in1, PLCOperand in2) {

    int input1, input2, des;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();
    des = dest.getAddress().getFileAddr();

    //IF EN
    //DEST = IN1+IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*2*/ banksel(input2);
    /*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input2, bsr, w));
    /*6*/ banksel(input1);
    /*8*/ asm.add(new ByteInstruction(Opcode.SUBWF, input1, bsr, w));
    /*10*/ banksel(des);
    /*12*/ asm.add(new ByteInstruction(Opcode.MOVWF, des, bsr, false));

    wordCount = 7;
    if (dest.getDataType() == PLCDataType.INT) {
        /*14*/ banksel(input2);
        /*16*/ asm.add(new ByteInstruction(Opcode.MOVF, input2 + 1, bsr,
w));
        /*18*/ asm.add(new ByteInstruction(Opcode.MOVWF, des + 1, bsr,
f));
        wordCount += 3;
    }
}

private void intAndInt(PLCOperand in1, PLCOperand in2) {
    int input1, input2, des;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();
    des = dest.getAddress().getFileAddr();

    //IF EN
    //DEST = IN1+IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));

```

```

/*2*/ banksel(input2);
/*4*/ asm.add(new ByteInstruction(Opcode.MOVF, input2, bsr, w));
/*6*/ banksel(input1);
/*8*/ asm.add(new ByteInstruction(Opcode.ADDWF, input1, bsr, w));
/*10*/ banksel(des);
/*12*/ asm.add(new ByteInstruction(Opcode.MOVWF, des, bsr, false));
wordCount = 7;
if (dest.getDataType() == PLCDataType.INT) {
    /*14*/ banksel(input2 + 1);
    /*16*/ asm.add(new ByteInstruction(Opcode.MOVF, input2 + 1, bsr,
w));
    /*18*/ banksel(input1 + 1);
    /*20*/ asm.add(new ByteInstruction(Opcode.SUBWFB, input1 + 1,
bsr, w));
    /*22*/ asm.add(new ByteInstruction(Opcode.MOVWF, des + 1, bsr,
false));
    wordCount += 5;
}
}

@Override
public List<Pic18Instruction> getPicAsm() {
    return asm;
}

@Override
public int getwordCount() {
    return wordCount;
}

private void banksel(int file) {
    byte bank = (byte) (file >> 8);
    asm.add(new LiteralInstruction(Opcode.MOVLB, bank));
}

private void branch(int currentAddress, int targetAddress) {
    //address = PC + 2 + 2n
    int branchAddress = targetAddress - currentAddress - 2;
    branchAddress /= 2;
    asm.add(new ControlInstruction(Opcode.BRA, branchAddress));
}
}

package com.vts.jplceditor.compiler.plc.instruction;

import com.vts.jplceditor.compiler.pic18.BitInstruction;
import com.vts.jplceditor.compiler.pic18.ControlInstruction;
import com.vts.jplceditor.compiler.pic18.LiteralInstruction;
import com.vts.jplceditor.compiler.pic18.Opcode;
import com.vts.jplceditor.compiler.pic18.Pic18Instruction;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.plc.PLCOperand;
import java.util.ArrayList;
import java.util.List;

/**

```

```

/*
 * @author Vusman
 */
public class PLC_Xic extends Pic18Mem implements PLCInstr {
    private final List<Pic18Instruction> asm;
    private final int bitAdress;
    private int org;
    private final byte EN = 0;
    private final byte ENO = 1;
    private int wordCount;
    private int count;
    private final PLCOOperand prevENO;

    public PLC_Xic(int org, List<PLCOOperand> parameters, PLCOOperand prevENO)
    {

        this.bitAdress = parameters.get(0).getAddress().getFileAddr();
        asm = new ArrayList<>();
        this.org = org;
        this.prevENO = prevENO;
        PLCOOperand in = parameters.get(2);
        checkENO();
        xic(in);
    }

    private void checkENO() {
        int code_start, clearENO;

        //First set the EN depending on the prev ENO
        if (prevENO == null) { //If no prev ENO, this is rung start
            banksel(bitAdress);
            asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN, bsr));
            wordCount = 2;
        } else { // Else set this EN depending on the prev ENO
            code_start = org + 18;
            clearENO = code_start + 12; //clear_ENO in instruction code
            int clrEN = org + 12;
            int enFile = prevENO.getAddress().getFileAddr();
            /*0*/ banksel(enFile);
            /*2*/ asm.add(new BitInstruction(Opcode.BTFSS, enFile, ENO,
bsr)); // If ENO set, set EN
            /*4*/ branch(org + 4, clrEN);
            /*6*/ banksel(bitAdress);
            /*8*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN,
bsr));
            /*10*/ branch(org+10,code_start);

            //clrEN address
            /*12*/ banksel(bitAdress);
            /*14*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, EN,
bsr));
            /*16*/ branch(org + 16, clearENO);
            wordCount = 9;
        }
        org += (wordCount * 2);
    }
}

```

```

private void xic(PLCOperand in) {
    int endif;
    int clear_ENO;
    byte bit = in.getAddress().getBitNo();
    int input = in.getAddress().getFileAddr();
    //IF EN
    //ENO = IN == FALSE
    //ENDIF
    clear_ENO = org + 12;
    endif = org+16;
    /*0*/ banksel(input);

    //if bit=FALSE set ENO
    /*2*/ asm.add(new BitInstruction(Opcode.BTFSC, input, bit, bsr));
    /*4*/ branch(org + 4, clear_ENO);
    /*6*/ banksel(bitAdress);
    /*8*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*10*/ branch(org + 10, endif);

    //clear_ENO address
    /*12*/ banksel(bitAdress);
    /*14*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, ENO, bsr));
    //endif address. Size is 20 bytes
    wordCount += 8;

}

@Override
public List<Pic18Instruction> getPicAsm() {
    return asm;
}

@Override
public int getwordCount() {
    return wordCount;
}

private void banksel(int file) {
    byte bank = (byte) (file >> 8);
    asm.add(new LiteralInstruction(Opcode.MOVLB, bank));
}

private void branch(int currentAddress, int targetAddress) {
    //address = PC + 2 + 2n
    int branchAddress = targetAddress - currentAddress - 2;
    branchAddress /= 2;
    asm.add(new ControlInstruction(Opcode.BRA, branchAddress));
}
}

```

```

package com.vts.jplceditor.compiler.plc.instruction;

import com.vts.jplceditor.compiler.pic18.BitInstruction;
import com.vts.jplceditor.compiler.pic18.ControlInstruction;
import com.vts.jplceditor.compiler.pic18.LiteralInstruction;
import com.vts.jplceditor.compiler.pic18.Opcode;
import com.vts.jplceditor.compiler.pic18.Pic18Instruction;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusman
 */
public class PLC_Xio extends Pic18Mem implements PLCInstr {

    private final List<Pic18Instruction> asm;
    private final int bitAdress;
    private int org;
    private final byte EN = 0;
    private final byte ENO = 1;
    private int wordCount;
    private int count;
    private final PLCOOperand prevENO;

    public PLC_Xio(int org, List<PLCOOperand> parameters, PLCOOperand prevENO)
    {

        this.bitAdress = parameters.get(0).getAddress().getFileAddr();
        asm = new ArrayList<>();
        this.org = org;
        this.prevENO = prevENO;
        PLCOOperand in = parameters.get(2);
        checkENO();
        xio(in);
    }

    private void checkENO() {
        int code_start, clearENO;

        //First set the EN depending on the prev ENO
        if (prevENO == null) { //If no prev ENO, this is rung start
            banksel(bitAdress);
            asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN, bsr));
            wordCount = 2;
        } else { // Else set this EN depending on the prev ENO
            code_start = org + 18;
            clearENO = code_start + 12; //clear_ENO in instruction code
            int clrEN = org + 12;
            int enFile = prevENO.getAddress().getFileAddr();
            /*0*/ banksel(enFile);
            /*2*/ asm.add(new BitInstruction(Opcode.BTFSS, enFile, ENO,
            bsr)); // If ENO set, set EN
            /*4*/ branch(org + 4, clrEN);
            /*6*/ banksel(bitAdress);
        }
    }
}

```

```

        /*8*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN,
bsr));
        /*10*/ branch(org + 10, code_start);

        //clrEN address
        /*12*/ banksel(bitAdress);
        /*14*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, EN,
bsr));
        /*16*/ branch(org + 16, clearENO);
        wordCount = 9;
    }
    org += (wordCount * 2);
}

private void xio(PLCOperand in) {
    int endif;
    int clear_ENO;
    byte bit = in.getAddress().getBitNo();
    int input = in.getAddress().getFileAddr();
    //IF EN
    //ENO = IN == TRUE
    //ENDIF
    clear_ENO = org + 12;
    endif = org + 16;
    /*0*/ banksel(input);

    //if bit=TRUE set ENO
    /*2*/ asm.add(new BitInstruction(Opcode.BTFSS, input, bit, bsr));
    /*4*/ branch(org + 4, clear_ENO);
    /*6*/ banksel(bitAdress);
    /*8*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*10*/ branch(org + 10, endif);

    //clear_ENO address
    /*12*/ banksel(bitAdress);
    /*14*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, ENO, bsr));
    //endif address. Size is 20 bytes
    wordCount += 8;
}

@Override
public List<Pic18Instruction> getPicAsm() {
    return asm;
}

@Override
public int getwordCount() {
    return wordCount;
}

private void banksel(int file) {
    byte bank = (byte) (file >> 8);
    asm.add(new LiteralInstruction(Opcode.MOVLB, bank));
}

```

```

private void branch(int currentAddress, int targetAddress) {
    //address = PC + 2 + 2n
    int branchAddress = targetAddress - currentAddress - 2;
    branchAddress /= 2;
    asm.add(new ControlInstruction(Opcode.BRA, branchAddress));
}
}

package com.vts.jplceditor.compiler.plc.instruction;

import com.vts.jplceditor.compiler.pic18.BitInstruction;
import com.vts.jplceditor.compiler.pic18.ByteInstruction;
import com.vts.jplceditor.compiler.pic18.ControlInstruction;
import com.vts.jplceditor.compiler.pic18.LiteralInstruction;
import com.vts.jplceditor.compiler.pic18.Opcode;
import com.vts.jplceditor.compiler.pic18.Pic18Instruction;
import com.vts.jplceditor.compiler.pic18.Pic18Mem;
import com.vts.jplceditor.compiler.plc.PLCDATAType;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Vusman
 */
public class PLC_Xor extends Pic18Mem implements PLCInstr {

    private final List<Pic18Instruction> asm;
    private final int bitAdress;
    private int org;
    private final byte EN = 0;
    private final byte ENO = 1;
    private int wordCount;
    private int count;
    private final PLCOOperand prevENO, dest;

    public PLC_Xor(int org, List<PLCOOperand> parameters, PLCOOperand prevENO)
    {

        this.bitAdress = parameters.get(0).getAddress().getFileAddr();
        asm = new ArrayList<>();
        this.org = org;
        PLCOOperand in1, in2;
        in1 = parameters.get(2);
        in2 = parameters.get(3);
        dest = parameters.get(4);
        this.prevENO = prevENO;

        if (in1.getDataType() == PLCDATAType.BYTE && in2.getDataType() ==
PLCDATAType.BYTE) {
            checkENO();
            byteAndByte(in1, in2);
        } else if (in1.getDataType() == PLCDATAType.BYTE && in2.getDataType() ==
PLCDATAType.INT) {
    }
}

```

```

        checkENO();
        byteAndInt(in1, in2);
    } else if (in1.getDataType() == PLCDataType.INT && in2.getDataType()
== PLCDataType.BYTE) {
        checkENO();
        byteAndInt(in2, in1);
    } else if (in1.getDataType() == PLCDataType.INT && in2.getDataType()
== PLCDataType.INT) {
        checkENO();
        intAndInt(in1, in2);
    }
}

private void checkENO() {
    int code_start;
    int clrEN = org + 8;
    //First set the EN depending on the prev ENO
    if (prevENO == null) { //If no prev ENO, this is rung start
        banksel(bitAdress);
        asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN, bsr));
        count = 2;
    } else { // Else set this EN depending on the prev ENO
        code_start = 9 + org;
        count = 9;
        int enFile = prevENO.getAddress().getFileAddr();
        /*0*/ banksel(enFile);
        /*1*/ asm.add(new BitInstruction(Opcode.BTFSS, enFile, ENO,
bsr)); // If ENO set, set EN
        /*2*/ asm.add(new ControlInstruction(Opcode.GOTO, clrEN));
        /*4*/ banksel(bitAdress);
        /*5*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, EN,
bsr));
        /*6*/ asm.add(new ControlInstruction(Opcode.GOTO, code_start));
    }

    //clrEN address
    /*8*/ asm.add(new BitInstruction(Opcode.BCF, bitAdress, EN,
bsr));
}
org += count;
}

private void byteAndByte(PLCOperand in1, PLCOperand in2) {
    int input1, input2, des;
    input1 = in1.getAddress().getFileAddr();
    input2 = in2.getAddress().getFileAddr();
    des = dest.getAddress().getFileAddr();

    //IF EN
    //DEST = IN1&IN2
    //ENDIF
    /*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
    /*1*/ banksel(input1);
    /*2*/ asm.add(new ByteInstruction(Opcode.MOVF, input1, bsr, w));
    /*3*/ banksel(input2);
    /*4*/ asm.add(new ByteInstruction(Opcode.XORWF, input2, bsr, w));
}

```

```

/*5*/ banksel(des);
/*6*/ asm.add(new ByteInstruction(Opcode.MOVWF, des, bsr, false));
wordCount = 7;
if (dest.getDataType() == PLCDataType.INT) {
    /*7*/ asm.add(new LiteralInstruction(Opcode.MOVLW, 0));
    /*8*/ asm.add(new ByteInstruction(Opcode.ADDWFC, des + 1, bsr,
w));
    wordCount += 2;
}
/*17*/ asm.add(new ControlInstruction(Opcode.NOP)); //endif
address
wordCount = 1 + count;
}

private void byteAndInt(PLCOperand in1, PLCOperand in2) {

int input1, input2, des;
input1 = in1.getAddress().getFileAddr();
input2 = in2.getAddress().getFileAddr();
des = dest.getAddress().getFileAddr();

//IF EN
//DEST = IN1&IN2
//ENDIF
/*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
/*1*/ banksel(input1);
/*2*/ asm.add(new ByteInstruction(Opcode.MOVF, input1, bsr, w));
/*3*/ banksel(input2);
/*4*/ asm.add(new ByteInstruction(Opcode.XORWF, input2, bsr, w));
/*5*/ banksel(des);
/*6*/ asm.add(new ByteInstruction(Opcode.MOVWF, des, bsr, false));

wordCount = 7;
if (dest.getDataType() == PLCDataType.INT) {
    /*8*/ banksel(input2+1);
    /*9*/ asm.add(new ByteInstruction(Opcode.MOVF, input2 + 1, bsr,
w));
    /*10*/ banksel(des+1);
    /*11*/ asm.add(new ByteInstruction(Opcode.MOVWF, des+1, bsr,
false));
    wordCount += 4;
}
/*17*/ asm.add(new ControlInstruction(Opcode.NOP)); //endif
address
wordCount = 1 + count;
}

private void intAndInt(PLCOperand in1, PLCOperand in2) {
int input1, input2, des;
input1 = in1.getAddress().getFileAddr();
input2 = in2.getAddress().getFileAddr();
des = dest.getAddress().getFileAddr();

//IF EN
//DEST = IN1&IN2

```

```

//ENDIF
/*0*/ asm.add(new BitInstruction(Opcode.BSF, bitAdress, ENO, bsr));
/*1*/ banksel(input1);
/*2*/ asm.add(new ByteInstruction(Opcode.MOVF, input1, bsr, w));
/*3*/ banksel(input2);
/*4*/ asm.add(new ByteInstruction(Opcode.XORWF, input2, bsr, w));
/*5*/ banksel(des);
/*6*/ asm.add(new ByteInstruction(Opcode.MOVWF, des, bsr, false));

wordCount = 7;
if (dest.getDataType() == PLCDataType.INT) {
    /*7*/ banksel(input1);
    /*8*/ asm.add(new ByteInstruction(Opcode.MOVF, input1+1, bsr,
w));
    /*9*/ banksel(input2);
    /*10*/ asm.add(new ByteInstruction(Opcode.XORWF, input2 + 1, bsr,
w));
    /*11*/ banksel(des);
    /*12*/ asm.add(new ByteInstruction(Opcode.MOVWF, des+1, bsr,
false));
    wordCount += 6;
}
/*17*/ asm.add(new ControlInstruction(Opcode.NOP));      //endif
address
wordCount = 1 + count;
}

@Override
public List<Pic18Instruction> getPicAsm() {
    return asm;
}

@Override
public int getwordCount() {
    return wordCount;
}

private void banksel(int file) {
    byte bank = (byte) (file >> 8);
    asm.add(new LiteralInstruction(Opcode.MOVLB, bank));
}
}

```

```

package com.vts.jplceditor.compiler.plc.instruction;

import com.vts.jplceditor.compiler.pic18.Pic18Instruction;
import java.util.List;

public interface PLCInstr{
    public List<Pic18Instruction> getPicAsm();
    public int getwordCount();
}

```

Appendix J: JPLC Editor Ladder Diagram layer classes code

```
package com.vts.jplceditor.ld;

import com.vts.jplceditor.ld.component.LDLine;
import com.vts.jplceditor.ld.component.LDRung;
import com.vts.jplceditor.ld.component.node.SNode;
import com.vts.jplceditor.ld.layout.RailLayout;
import com.vts.jplceditor.ld.layout.Size;
import java.util.ArrayList;
import java.util.List;
import javafx.scene.Group;
import javafx.scene.layout.Pane;

/**
 *
 * @author Vusumuzi_Tshabangu
 */
public class LadderDiagram extends Pane {

    private final int width = 640;
    private final int height = 720;
    private RailLayout rungLayout;
    private List<LDRung> rungList;
    private LDLine sRail, tRail;
    private Group rungs;
    private Group nodes;
    private List<SNode> sourceNodes;

    /**
     *
     */
    public LadderDiagram() {
        super.setWidth(width);
        super.setHeight(height);
        sourceNodes = new ArrayList<>();
        rungs = new Group();
        nodes = new Group();
        getChildren().add(rungs);
        getChildren().add(nodes);
        rungList = new ArrayList<>();
        createRail();
        createDefaultRung();
    }

    public LadderDiagram(List<LDRung> rungList, List<SNode> sourceNodes) {
        this.rungList = rungList;
        this.sourceNodes = sourceNodes;
        super.setWidth(width);
        super.setHeight(height);
        rungs = new Group();
        nodes = new Group();
        getChildren().add(rungs);
        getChildren().add(nodes);
        createRail();
        for (LDRung rung : rungList) {
```

```

        rungs.getChildren().add(rung);
        rungList.add(rung);
        getChildren().add(rung.getInstructions());
    }
    for (SNode snode : sourceNodes) {
        nodes.getChildren().add(snode);
    }
}

/**
 *
 */
public void addRung(int position) {
    rungLayout.addNewSourceNode();
    List<SNode> sNodes = rungLayout.getSourceNodes();
    int count = sNodes.size();
    SNode sNode = rungLayout.getSourceNodes().get(count - 1);
    LDRung r = new LDRung(sNode, new Size(width, RailLayout.RUNG_SIZE),
2);
    nodes.getChildren().add(sNode);
    rungList.add( r );
    rungs.getChildren().add(r);
    getChildren().add(r.getInstructionGroup());

    // for(int i=0;i<count;i++){
    //     rungList.get(i).relocateToPoint(sNodes.get(i).getLocation());
    // }
}

private void createRail() {
    rungLayout = new RailLayout(width, height);
    sRail = new LDLine(rungLayout.getRailStart(),
rungLayout.getRailEnd());
    sRail.setStrokeWidth(4);
    getChildren().add(sRail);
}

public List<SNode> getSourceNodes() {
    return sourceNodes;
}

private void createDefaultRung() {
    SNode point = rungLayout.getSourceNodes().get(0);
    LDRung r = new LDRung(point, new Size(width, RailLayout.RUNG_SIZE),
2);
    nodes.getChildren().add(r.getsNode());
    rungList.add(r);
    rungs.getChildren().add(r);
    getChildren().add(r.getInstructions());
}

/**
 *
 * @return
 */
public List<LDRung> getRungList() {
    return rungList;
}

```

```
}

}

package com.vts.jplceditor.ld.component;

/**
 *
 * @author Vusivus
 */
public interface LDComponent {
    public void select();
    public void deselect();
}

package com.vts.jplceditor.ld.component;

import com.vts.jplceditor.compiler.plc.PLCOperator;
import com.vts.jplceditor.compiler.plc.PLCOperatorDecoder;
import com.vts.jplceditor.ld.component.node.INodeChangeListener;
import com.vts.jplceditor.ld.component.node.NodeType;
import com.vts.jplceditor.ld.instruction.Instruction;
import com.vts.jplceditor.ld.instruction.InstructionUtil;
import com.vts.jplceditor.ld.layout.Size;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import javafx.event.EventHandler;
import javafx.geometryInsets;
import javafx.geometry.Point2D;
import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.MouseEvent;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.CornerRadii;
import javafx.scene.layout.Region;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;

/**
 *
 * @author Vusivus
 */
public class LDInstruction extends Region implements LDNode {

    private Point2D startPoint, endPoint, cornerPoint;
    private EventHandler<DragEvent> dragEntered;
    private EventHandler<DragEvent> dragExited;
    private EventHandler<DragEvent> dragDropped;
    private EventHandler<DragEvent> dragOver;
    private EventHandler<MouseEvent> mouseClicked;
```

```

private final BackgroundFill fill, fillDragOver, fillDragDropped;
private Size size = new Size(40, 20);
private boolean hasInstruction, selected;
private final List<INodeChangeListener> listeners;
private int inDegree, outDegree;
private Instruction instruction;
private Rectangle selection;

/**
 * <p>
 * creates an InstructionNode block centered at this point</p>
 *
 * @param startPoint
 */
public LDInstruction(Point2D startPoint) {
    this.startPoint = startPoint;

    //Creates the default blue color of the square block
    fill = new BackgroundFill(Color.BLUE, CornerRadii.EMPTY,
    Insets.EMPTY);
    fillDragOver = new BackgroundFill(Color.RED, CornerRadii.EMPTY,
    Insets.EMPTY);
    fillDragDropped = new BackgroundFill(Color.WHITE, CornerRadii.EMPTY,
    Insets.EMPTY);
    Background background = new Background(fill);
    setBackground(background);

    endPoint = startPoint.add(size.getWidth(), 0);
    cornerPoint = startPoint.subtract(0, size.getHeight() / 2);
    setLayoutX(cornerPoint.getX());
    setLayoutY(cornerPoint.getY());
    setPrefWidth(size.getWidth());
    setPrefHeight(size.getHeight());
    //Assigns a unique ID to this Node
    setId(UUID.randomUUID().toString());
    listeners = new ArrayList<>();
    createDragHandlers();
}

/**
 * An INode change listener is notified every time this node is assigned
 * an
 * Instruction
 *
 * @param listener
 */
public void registerChangeListener(INodeChangeListener listener) {
    listeners.add(listener);
}

/**
 * Assigns an Instruction to this INode
 *
 * @param operator
 */
public void setInstruction(PLCOperator operator) {
}

```

```
if (!getChildren().isEmpty()) {
    getChildren().remove(0);
}

if (operator != PLCOperator.NULL) {
    instruction = InstructionUtil.getInstruction(operator.name());
    AnchorPane node = instruction.getInstance();
    size.setWidth(node.getPrefWidth());
    size.setHeight(node.getPrefHeight());
    setPrefWidth(size.getWidth());
    setPrefHeight(size.getHeight());

    getChildren().add(0, node);

    relocateToPoint(startPoint);
    hasInstruction = true;

    if (listeners.size() > 0) {
        for (int i = 0; i < listeners.size(); i++) {
            listeners.get(i).onInstructionAdded(getId());
        }
    }
}

}

public void select() {
    if (!selected && hasInstruction) {
        double x, y;
        x = cornerPoint.subtract(10, 10).getX();
        y = cornerPoint.subtract(10, 10).getY();
        selection = new Rectangle(x, y, Color.TRANSPARENT);
        selection.setStroke(Color.BLUE);
        selection.setWidth(size.getWidth() + 10);
        selection.setHeight(size.getHeight() + 10);
        getChildren().add(1, selection);
        selected = true;
    }
}

public void deselect() {
    if (selected) {
        getChildren().remove(1);
        selected = false;
    }
}

public void removeInstruction() {
    if (!getChildren().isEmpty()) {
        getChildren().remove(0);
        size.setHeight(20);
        size.setWidth(40);
        setPrefWidth(40);
        setPrefHeight(20);
        relocateToPoint(startPoint);
    }
}
```

```
        setBackground(new Background(fill));
        instruction = null;
        hasInstruction = false;
    }

    if (listeners.size() > 0) {
        for (int i = 0; i < listeners.size(); i++) {
            listeners.get(i).onInstructionRemoved(getId());
        }
    }
}

public Point2D getStartPoint() {
    return startPoint;
}

public Point2D getEndPoint() {
    return endPoint;
}

public void relocateToPoint(Point2D point) {
    startPoint = point;
    endPoint = startPoint.add(size.getWidth(), 0);
    cornerPoint = startPoint.subtract(0, size.getHeight() / 2);
    relocate(cornerPoint.getX(), cornerPoint.getY());
}

public void setSize(Size size) {
    this.size = size;
    endPoint = startPoint.add(size.getWidth(), 0);
    cornerPoint = startPoint.subtract(0, size.getHeight() / 2);
}

public Size getSize() {
    return size;
}

public Point2D getCornerPoint() {
    return cornerPoint;
}

public boolean hasInstruction() {
    return hasInstruction;
}

private void createDragHandlers() {
    dragEntered = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                if (getChildren().isEmpty()) {
                    setBackground(new Background(fillDragOver));
                }
            }
            event.consume();
        }
    };
}
```

```

};

dragExited = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            if (getChildren().isEmpty()) {
                setBackground(new Background(fill));
            }
        }
        event.consume();
    }
};

dragOver = new EventHandler<DragEvent>() {
    public void handle(DragEvent event) {

        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            /* allow for both copying and moving, whatever user
chooses */
            event.acceptTransferModes(TransferMode.COPY_OR_MOVE);
        }

        event.consume();
    }
};

dragDropped = new EventHandler<DragEvent>() {
    public void handle(DragEvent event) {

        Dragboard db = event.getDragboard();
        if (db.hasString()) {
            String name = db.getString();
            PLCOperator i = PLCOperatorDecoder.fromID(name);
            if (i != null) {
                setBackground(new Background(fillDragDropped));
                setInstruction(i);
            }

            event.setDropCompleted(true);
            event.consume();
        }
    }
};

mouseClicked = new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event) {
        for (INodeChangeListener listener : listeners) {
            listener.onInstructionClicked(getId(), event);
        }
    }
};
setOnDragEntered(dragEntered);
setOnDragExited(dragExited);

```

```
        setOnDragDropped(dragDropped);
        setOnDragOver(dragOver);
        setOnMouseClicked(mouseClicked);
    }

    public Instruction getInstruction() {
        return instruction;
    }

    @Override
    public void setInDegree(int inDegree) {

    }

    @Override
    public void setOutDegree(int outDegree) {

    }

    @Override
    public int getInDegree() {
        return 1;
    }

    @Override
    public int getOutDegree() {
        return 0;
    }

    @Override
    public NodeType getNodeType() {
        return NodeType.Instruction;
    }

}

package com.vts.jplceditor.ld.component;

import com.vts.jplceditor.ld.layout.Size;
import java.util.ArrayList;
import java.util.List;
import javafx.geometry.Point2D;
import javafx.scene.Group;

/**
 *
 * @author Vusivus
 */
public class LDInstructionManager extends Group {

    private final List<LDInstruction> instructionList;
    public static int SPACING = 100;
    private Point2D sPoint;
    private short minInstructionCount = 2;

    /**

```

```

*
 * @param origin
 */
public LDInstructionManager(Point2D origin) {
    instructionList = new ArrayList<>();
    sPoint = origin;
}

/**
 * Highlights the provided instruction
 *
 * @param position
 */
public void selectOnly(int position) {
    if (contained(position)) {
        for (LDInstruction i : instructionList) {
            i.deselect();
        }
        instructionList.get(position).select();
    }
}

public Point2D startPointOf(String nodeID) {
    int index = instructionIndex(nodeID);
    if (contained(index)) {
        return instructionList.get(index).getStartPoint();
    } else {
        return null;
    }
}

public Point2D endPointOf(String nodeID) {
    int index = instructionIndex(nodeID);
    if (contained(index)) {
        return instructionList.get(index).getEndPoint();
    } else {
        return null;
    }
}

public Size SizeOf(String nodeID) {
    int index = instructionIndex(nodeID);
    if (contained(index)) {
        return instructionList.get(index).getSize();
    } else {
        return null;
    }
}

/**
 *
 * @param origin
 * @param instructionList
 */
public LDInstructionManager(Point2D origin, List<LDInstruction>
instructionList) {
    this.instructionList = new ArrayList<>();
}

```

```

        for (int i = 0; i < instructionList.size(); i++) {
            addInstruction(i, instructionList.get(i));
        }
        sPoint = origin;
    }

    /**
     *
     * @param position
     * @param inode
     */
    public void addInstruction(int position, LDInstruction inode) {
        instructionList.add(position, inode);
        getChildren().add(position, inode);
    }

    /**
     * Removes the instruction at the position provided
     *
     * @param position
     */
    public void removeInstruction(int position) {
        if (instructionList.size() > minInstructionCount) { //if we have more
than 2 instructions
            if (position < instructionList.size()) {
                instructionList.remove(position);
                getChildren().remove(position);
                shiftAfterRemove(position); // shift instructions to the
left after remove
            } else {
                System.out.println("Could not remove instruction at position
" + position
                           + " Instruction index is not within the bounds " +
instructionList.size());
            }
        } else {
            System.out.println("Could not remove instruction at position " +
position);
        }
    }

    /**
     *
     * @param id
     * @return
     */
    public int instructionIndex(String id) {
        int index = -1;
        String identity;
        for (int i = 0; i < instructionList.size(); i++) {
            identity = instructionList.get(i).idProperty().get();
            if (identity.equals(id)) {
                index = i;
                break;
            }
        }
        return index;
    }
}

```

```

}

/**
 *
 * @param amount
 * @param start
 */
public void shiftInstrX(double amount, int start) {
    if (contained(start)) {
        Point2D p;
        for (int i = start; i < instructionList.size(); i++) {
            p = instructionList.get(i).getStartPoint();
            instructionList.get(i).relocateToPoint(p.add(amount, 0));
        }
    } else {
        System.out.println("Cannot shift uncontained position " + start);
    }
}

public void shiftInstrY(double amount, int start) {
    if (contained(start)) {
        Point2D p;
        for (int i = start; i < instructionList.size(); i++) {
            p = instructionList.get(i).getStartPoint();
            instructionList.get(i).relocateToPoint(p.add(0, amount));
        }
    } else {
        System.out.println("Cannot shift uncontained position " + start);
    }
}

public Point2D getTPoint() {
    return instructionList.get(count() - 1).getEndPoint();
}

/**
 * <p>
 * Calculates a point to the right where a new instruction node can. be
 * placed.</p>
 *
 * @param point The index of the instruction to add from
 * @param tPoint The termination point of the rung
 * @return
 */
public Point2D nextFreeSlot(int point, Point2D tPoint) {
    Point2D p = null;
    if (point < instructionList.size() - 1) {
        Point2D p1, p2;
        p1 = instructionList.get(point).getEndPoint();
        p2 = instructionList.get(point + 1).getStartPoint();
        double dist = Math.abs(p1.getX() - p2.getX());
        double min = 2 * SPACING + 40;
        if (dist < SPACING) {
            shiftInstrX(min - dist, point + 1);
        }
        double x, y;
        x = p1.getX() + SPACING;
    }
}

```

```

        y = p1.getY();
        p = new Point2D(x, y);
    } else if (point == instructionList.size() - 1) {
        Point2D p1;
        p1 = instructionList.get(point).getEndPoint();
        double dist = Math.abs(p1.getX() - tPoint.getX());
        double min = 2 * SPACING + 40;
        if (dist < min) {
            shiftInstrX(min - dist, point + 1);
        }
        double x, y;
        x = p1.getX() + SPACING;
        y = p1.getY();
        p = new Point2D(x, y);
    }
    return p;
}

/**
 *
 * @return
 */
public List<LDInstruction> getInstructionList() {
    return instructionList;
}

/**
 *
 * @return
 */
public int count() {
    return instructionList.size();
}

/**
 *
 * @param index
 * @return
 */
public boolean contained(int index) {
    return index < instructionList.size();
}

private void shiftAfterRemove(int position) {
    if (contained(position)) {
        double p1, p2;
        if (position > 0) {

            p1 = instructionList.get(position - 1).getEndPoint().getX();
        } else {
            p1 = sPoint.getX();
        }
        p2 = instructionList.get(position).getStartPoint().getX();
        double dist = p2 - p1;
        if (dist > SPACING) {
            shiftInstrX(SPACING - dist, position);
        }
    }
}

```

```
        } else {
            System.out.println("Cannot shift instructions about uncontained
position " + position);
        }
    }

    public void setMinInstructionCount(short minInstructionCount) {
        this.minInstructionCount = minInstructionCount;
    }
}

package com.vts.jplceditor.ld.component;

import javafx.geometry.Point2D;
import javafx.scene.shape.Line;

/**
 *
 * @author Vusumuzi_Tshabangu
 */
public class LDJoin{
    private Line sLine, hLine;
    private Point2D startPoint, endPoint;

    public LDJoin(Point2D startPoint, Point2D endPoint) {
        this.startPoint = startPoint;
        this.endPoint = endPoint;

        buildHLine();
        buildSLine();
    }

    private void buildHLine() {
        hLine = new Line();
        hLine.setStrokeWidth(2);
        hLine.setStartX(startPoint.getX());
        hLine.setEndX(endPoint.getX());
        hLine.setStartY(endPoint.getY());
        hLine.setEndY(endPoint.getY());
    }

    private void buildSLine() {
        sLine = new Line();
        sLine.setStrokeWidth(2);
        sLine.setStartX(startPoint.getX());
        sLine.setEndX(startPoint.getX());
        sLine.setStartY(startPoint.getY());
        sLine.setEndY(endPoint.getY());
    }

    public void setStartPoint(Point2D startPoint) {
        this.startPoint = startPoint;
        sLine.setStartX(startPoint.getX());
        sLine.setEndX(startPoint.getX());
        sLine.setStartY(startPoint.getY());
    }
}
```

```
        hLine.setStartX(startPoint.getX());
    }

    public void setEndPoint(Point2D endPoint) {
        this.endPoint = endPoint;
        sLine.setEndY(endPoint.getY());
        hLine.setEndX(endPoint.getX());
        hLine.setStartY(endPoint.getY());
        hLine.setEndY(endPoint.getY());
    }

    public Line getsLine() {
        return sLine;
    }

    public Line gethLine() {
        return hLine;
    }

}

package com.vts.jplceditor.ld.component;

import java.util.HashMap;
import java.util.UUID;
import javafx.geometry.Point2D;
import javafx.scene.Node;
import javafx.scene.shape.Line;

/**
 *Represents a line on a ladder diagram
 * @author Vusumuzi_Tshabangu
 */
public class LDLine extends Line{

    /**
     *
     * @param startPoint
     * @param endPoint
     */
    public LDLine(Point2D startPoint, Point2D endPoint) {
        setStartX(startPoint.getX());
        setStartY(startPoint.getY());
        setEndX(endPoint.getX());
        setEndY(endPoint.getY());
        setStrokeWidth(2);
        setId(UUID.randomUUID().toString());
    }

    /**
     *Relocates the starting point
     * @param startPoint
     */
    public void setStartPoint(Point2D startPoint){
        setStartX(startPoint.getX());
        setStartY(startPoint.getY());
    }
}
```

```
}

/**
 *Relocates the end point
 * @param endPoint
 */
public void setEndPoint(Point2D endPoint){
    setEndX(endPoint.getX());
    setEndY(endPoint.getY());
}

}

package com.vts.jplceditor.ld.component;

import javafx.geometry.Point2D;
import javafx.scene.Group;

*
*
* @author Vusivus

public class LDLineManager extends Group{

    public LDLineManager() {

    }

    public void addLine(Point2D sPoint, Point2D tPoint){
        getChildren().add(new LDLine(sPoint,tPoint));
    }

    public void relocateLine(String id,Point2D p1,Point2D p2){
        LDLine temp;
        for(int i=0;i<= getChildren().size();i++){
            if(getChildren().get(i).getId() == null ? id == null :
getChildren().get(i).getId().equals(id)){
                temp = (LDLine) getChildren().get(i);
                temp.setStartPoint(p1);
                temp.setEndPoint(p2);
            }
        }
    }

    public void clearAll(){
        getChildren().clear();
    }
}

package com.vts.jplceditor.ld.component;

import com.vts.jplceditor.ld.component.node.NodeType;
```

```
/**  
 *  
 * @author Vusivus  
 */  
public interface LDNode {  
    public void setInDegree(int inDegree);  
    public void setOutDegree(int outDegree);  
    public int getInDegree();  
    public int getOutDegree();  
    public NodeType getNodeType();  
}  
  
package com.vts.jplceditor.ld.component;  
  
import java.util.List;  
  
/**  
 *  
 * @author Vusivus  
 */  
public interface LDParent {  
    public void addChild(LDNode child);  
    public void atChildAt(int index,LDNode child);  
    public List<LDNode> getChildren();  
}  
  
package com.vts.jplceditor.ld.component;  
  
import com.vts.jplceditor.ld.component.node.INodeChangeListener;  
import com.vts.jplceditor.ld.component.node.SNode;  
import com.vts.jplceditor.ld.layout.Size;  
import java.util.ArrayList;  
import java.util.List;  
import javafx.event.ActionEvent;  
import javafx.event.EventHandler;  
import javafx.geometry.Point2D;  
import javafx.scene.Group;  
import javafx.scene.Node;  
import javafx.scene.control.ContextMenu;  
import javafx.scene.control.MenuItem;  
import javafx.scene.input.MouseButton;  
import javafx.scene.input.MouseEvent;  
import javafx.scene.layout.Region;  
  
/**  
 *  
 * @author Vusivus  
 */  
public class LDRung extends Region implements INodeChangeListener{  
  
    private Point2D sPoint, tPoint, cornerPoint;  
    private LDLineManager rungLines;  
    private LDInstructionManager instructions;  
    private Group branches;
```

```

private Group instructionGroup;
private Size size;
private final SNode sNode;
private ContextMenu menu;
private String selected;

/**
 *
 * @param sNode The source node where this rung will start from
 * @param size the dimensions of the rung
 */
public LDRung(SNode sNode, Size size, int iCount) {
    this.sNode = sNode;
    this.size = size;
    sPoint = sNode.getLocation();
    tPoint = sPoint.add(size.getWidth(), 0);
    cornerPoint = sPoint.subtract(0, size.getHeight() / 2);
    setPrefWidth(size.getWidth());
    setPrefHeight(size.getHeight());
    rungLines = new LDLineManager();
    instructions = new LDInstructionManager(sPoint);
    branches = new Group();
    instructionGroup = new Group();
    instructionGroup.getChildren().add(instructions);
    getChildren().add(branches);
    createDefaultRung(iCount);
    buildMenu();
}

public LDRung(LDInstructionManager instructions, Size size, SNode sNode)
{
    this.instructions = instructions;
    this.size = size;
    this.sNode = sNode;
    sPoint = sNode.getLocation();
    tPoint = sPoint.add(size.getWidth(), 0);
    cornerPoint = sPoint.subtract(0, size.getHeight() / 2);
    setPrefWidth(size.getWidth());
    setPrefHeight(size.getHeight());
    rungLines = new LDLineManager();
    getChildren().add(rungLines);
    redrawRung();
    buildMenu();
}

/**
 *
 * @return
 */
public SNode getsNode() {
    return sNode;
}

public LDInstructionManager getInstructions() {
    return instructions;
}

```

```

public void removeSelected() {
    int index = instructions.instructionIndex(selected);
    if (index >= 0) {
        LDInstruction i = instructions.getInstructionList().get(index);
        i.removeInstruction();
    }
}

private void createDefaultRung(int iCount) {
    getChildren().add(rungLines);
    //getChildren().add(instructions);
    if (iCount == 2) {
        double dist = tPoint.getX() - sPoint.getX();
        LDInstruction i1 = new LDInstruction(sPoint.add(dist * 0.25f,
0));
        LDInstruction i2 = new LDInstruction(sPoint.add(dist * 0.75f,
0));
        i1.registerChangeListener(this);
        i2.registerChangeListener(this);
        instructions.addInstruction(0, i1);
        instructions.addInstruction(1, i2);
        sNode.addChild(i1);
        sNode.addChild(i2);

        rungLines.addLine(sPoint, i1.getStartPoint());
        rungLines.addLine(i1.getEndPoint(), i2.getStartPoint());
        rungLines.addLine(i2.getEndPoint(), tPoint);
    } else if (iCount == 1) {
        double dist = tPoint.getX() - sPoint.getX();
        LDInstruction i1 = new LDInstruction(sPoint.add(dist * 0.33f,
0));
        i1.registerChangeListener(this);
        instructions.setMinInstructionCount((short) iCount);
        instructions.addInstruction(0, i1);
        sNode.addChild(i1);

        rungLines.addLine(sPoint, i1.getStartPoint());
        rungLines.addLine(i1.getEndPoint(), tPoint);
    }
}

private void redrawRung() {
    rungLines.clearAll();
    rungLines.addLine(sPoint,
instructions.getInstructionList().get(0).getStartPoint());
    Point2D p1, p2;
    int count = instructions.count();
    for (int i = 0; i < count; i++) {
        if (i != instructions.count() - 1) {
            p1 = instructions.getInstructionList().get(i).getEndPoint();
            p2 = instructions.getInstructionList().get(i +
1).getStartPoint();
            rungLines.addLine(p1, p2);
        }
    }
}

```

```

        p1 = instructions.getInstructionList().get(count - 1).getEndPoint();
        rungLines.addLine(p1, tPoint);
    }

    /**
     *
     * Invoked whenever an INode is updated with an instruction.
     * <p>
     * adds a blank instruction node</p>
     *
     * @param nodeID
     */
    @Override
    public void onInstructionAdded(String nodeID) {
        int index = instructions.instructionIndex(nodeID);
        if (index > -1) {
            Point2D p = instructions.nextFreeSlot(index, tPoint);
            LDInstruction in = new LDInstruction(p);
            in.registerChangeListener(this);
            p = tPoint.add(0, 0);
            instructions.addInstruction(index + 1, in);
            tPoint = instructions.getTPoint();
            sNode.addChild(in);
            redrawRung();
        }
    }

    public Group getBranches() {
        return branches;
    }

    /**
     *
     * @param nodeID
     */
    @Override
    public void onInstructionRemoved(String nodeID) {
        int index = instructions.instructionIndex(nodeID);

        if (index > -1) {
            Point2D p = tPoint.add(0, 0);
            instructions.removeInstruction(index);
            tPoint = instructions.getTPoint();
            redrawRung();
        } else {
            System.out.println("Failed to remove instruction with ID: " +
nodeID
                + "It's index " + index + " is out of bounds.");
        }
    }

    public void cleanUp() {
        for (int i = 0; i < instructions.getInstructionList().size(); i++) {
            if (!instructions.getInstructionList().get(i).hasInstruction()) {
                instructions.removeInstruction(i);
                redrawRung();
            }
        }
    }
}

```

```

        }

    }

    public void relocateToPoint(Point2D point) {
        instructions.shiftInstrY(point.getY()-sPoint.getY(), 0);
        sPoint = point;
        tPoint = sPoint.add(size.getWidth(), 0);
        cornerPoint = sPoint.subtract(0, size.getHeight() / 2);
        relocate(cornerPoint.getX(), cornerPoint.getY());
        redrawRung();
    }

    public void setSize(Size size) {
        this.size = size;
        tPoint = sPoint.add(size.getWidth(), 0);
        cornerPoint = sPoint.subtract(0, size.getHeight() / 2);
    }

    public Size getSize() {
        return size;
    }

    public Point2D getStartPoint() {
        return sPoint;
    }

    public Point2D getEndPoint() {
        return tPoint;
    }

    public Point2D getCornerPoint() {
        return cornerPoint;
    }

    public List<LDInstruction> getInstructionList() {
        return instructions.getInstructionList();
    }

    @Override
    public void onInstructionClicked(String nodeId, MouseEvent event) {
        selected = nodeId;
        int index = instructions.instructionIndex(selected);
        if (event.getButton() == MouseButton.PRIMARY) {
            instructions.selectOnly(index);
        } else if (event.getButton() == MouseButton.SECONDARY) {
            Node node = instructions.getInstructionList().get(index);
            menu.show(node, event.getScreenX(), event.getSceneY());
        }
    }

    private void buildMenu() {
        menu = new ContextMenu();

        MenuItem delete = new MenuItem("delete");
        delete.setOnAction(new EventHandler<ActionEvent>() {
            @Override

```

```

        public void handle(ActionEvent event) {
            int index = instructions.instructionIndex(selected);
            if (index >= 0) {
                instructions.removeInstruction(index);
                redrawRung();
            }
        }
    });
MenuItem remove = new MenuItem("remove instruction");
remove.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        int index = instructions.instructionIndex(selected);
        if (index >= 0) {
            LDInstruction i =
instructions.getInstructionList().get(index);
            i.removeInstruction();
        }
    }
});
menu.getItems().addAll(delete, remove);
}
public Group getInstructionGroup() {
    return instructionGroup;
}

}

package com.vts.jplceditor.ld.component.node;

import com.vts.jplceditor.ld.component.LDNode;
import com.vts.jplceditor.ld.component.LDParent;
import java.util.ArrayList;
import java.util.List;
import javafx.geometry.Point2D;
import javafx.scene.paint.Color;

/**
 *
 * @author Vusivus
 */
public class BNode extends Vertex implements LDNode, LDParent{

    private final List<LDNode> children;
    private int inDegree,outDegree;

    public BNode(Point2D location) {
        super(location);
        setNodeType(NodeType.Branch);
        setFill(Color.BLUE);
        children = new ArrayList<LDNode>();
    }
}

```

```
}

@Override
public void setInDegree(int inDegree) {
    this.inDegree = inDegree;
}

@Override
public void setOutDegree(int outDegree) {
    this.outDegree = outDegree;
}

@Override
public int getInDegree() {
    return inDegree;
}

@Override
public int getOutDegree() {
    return outDegree;
}

@Override
public NodeType getNodeType() {
    return NodeType.Branch;
}

@Override
public void addChild(LDNode child) {
    children.add(child);
}

@Override
public void atChildAt(int index, LDNode child) {
    children.add(index, child);
}

@Override
public List<LDNode> getChildren() {
    return children;
}

}

package com.vts.jplceditor.ld.component.node;

import javafx.scene.input.MouseEvent;

/**
 *
 * @author Vusivus
 */
public interface INodeChangeListener {
    public void onInstructionAdded(String nodeID);
    public void onInstructionRemoved(String nodeID);
    public void onInstructionClicked(String nodeId, MouseEvent event);
}
```

}

```
package com.vts.jplceditor.ld.component.node;

import com.vts.jplceditor.ld.component.LDNode;
import javafx.geometry.Point2D;
import javafx.scene.paint.Color;

/**
 *
 * @author Vusivus
 */
public class JNode extends Vertex implements LDNode {

    private int inDegree, outDegree;

    public JNode(Point2D location) {
        super(location);
        setNodeType(NodeType.Join);
        setFill(Color.GREEN);
    }

    @Override
    public void setInDegree(int inDegree) {
        this.inDegree = inDegree;
    }

    @Override
    public void setOutDegree(int outDegree) {
        this.outDegree = outDegree;
    }

    @Override
    public int getInDegree() {
        return inDegree;
    }

    @Override
    public int getOutDegree() {
        return outDegree;
    }

    @Override
    public NodeType getNodeType() {
        return NodeType.Join;
    }
}

package com.vts.jplceditor.ld.component.node;

/**
 *
 * @author Vusivus
 */
public enum NodeType {
```

```
Instruction,Source,Termination,Branch,Join,FunctionBlock  
}
```

```
package com.vts.jplceditor.ld.component.node;  
  
import com.vts.jplceditor.ld.component.LDNode;  
import com.vts.jplceditor.ld.component.LDParent;  
import java.util.ArrayList;  
import java.util.List;  
import javafx.geometry.Point2D;  
import javafx.scene.paint.Color;  
  
/**  
 *  
 * @author Vusivus  
 */  
public class SNode extends Vertex implements LDNode, LDParent{  
  
    private int inDegree,outDegree;  
    private final List<LDNode> children;  
  
    public SNode(Point2D location) {  
        super(location);  
        setNodeType(NodeType.Source);  
        setFill(Color.RED);  
        children = new ArrayList<>();  
    }  
  
    @Override  
    public void setInDegree(int inDegree) {  
        this.inDegree = inDegree;  
    }  
  
    @Override  
    public void setOutDegree(int outDegree) {  
        this.outDegree = outDegree;  
    }  
  
    @Override  
    public int getInDegree() {  
        return inDegree;  
    }  
  
    @Override  
    public int getOutDegree() {  
        return outDegree;  
    }  
  
    @Override  
    public NodeType getNodeType(){  
        return NodeType.Source;  
    }  
  
    @Override  
    public void addChild(LDNode child) {
```

```
        children.add(child);
    }

@Override
public void atChildAt(int index, LDNode child) {
    children.add(index, child);
}

@Override
public List<LDNode> getChildren() {
    return children;
}
}

package com.vts.jplceditor.ld.component.node;

import com.vts.jplceditor.ld.component.LDNode;
import javafx.geometry.Point2D;
import javafx.scene.paint.Color;

/**
 *
 * @author Vusivus
 */
public class TNode extends Vertex implements LDNode{

    private int inDegree,outDegree;
    public TNode(Point2D location) {
        super(location);
        setNodeType(NodeType.Termination);
        setFill(Color.RED);
    }

    @Override
    public void setInDegree(int inDegree) {
        this.inDegree = inDegree;
    }

    @Override
    public void setOutDegree(int outDegree) {
        this.outDegree = outDegree;
    }

    @Override
    public int getInDegree() {
        return inDegree;
    }

    @Override
    public int getOutDegree() {
        return outDegree;
    }

    @Override
    public NodeType getNodeType(){
        return NodeType.Termination;
    }
}
```

```
}

package com.vts.jplceditor.ld.component.node;

import javafx.geometry.Point2D;
import javafx.scene.shape.Circle;

/**
 *
 * @author Vusivus
 */
public class Vertex extends Circle{
    Point2D location;
    NodeType nodeType;

    public Vertex(Point2D location) {
        super(location.getX(),location.getY(),6);
        this.location = location;
    }

    public Point2D getLocation() {
        return location;
    }

    public void setLocation(Point2D location) {
        this.location = location;
        setCenterX(location.getX());
        setCenterY(location.getY());
    }

    public NodeType getNodeType() {
        return nodeType;
    }

    public void setNodeType(NodeType.nodeType) {
        this.nodeType = nodeType;
    }

}

package com.vts.jplceditor.ld.layout;

import com.vts.jplceditor.ld.component.node.SNode;
import java.util.ArrayList;
import java.util.List;
import javafx.geometry.Point2D;

/**
 *
 * @author Vusumuzi_Tshabangu
 */
```

```
/*
public class RailLayout {
    private List <SNode> sourceNodes;
    private double width;
    private double height;
    private double railHeight,rungWidth;
    private Point2D railStart, railEnd;
    public static final int RUNG_SIZE = 150;

    public RailLayout(double width, double height) {
        this.width = width;
        this.height = height;
        createRails();
        createSourceNodes();
    }

    public double getWidth() {
        return width;
    }

    public void setWidth(double width) {
        this.width = width;
    }

    public double getHeight() {
        return height;
    }

    public void setHeight(double height) {
        this.height = height;
    }

    private void createRails() {
        double x = 0.05*width;
        double y1 = 0.05*height;
        double y2 = 0.95*height;
        railStart = new Point2D(x,y1);
        railEnd = new Point2D(x,y2);
        railHeight = y2-y1;
        rungWidth = 0.1*width;
    }

    public Point2D getRailStart() {
        return railStart;
    }

    public Point2D getRailEnd() {
        return railEnd;
    }

    private void createSourceNodes() {
        sourceNodes = new ArrayList<>();
        addNewSourceNode();
    }

    public List<SNode> getSourceNodes() {
```

```
        return sourceNodes;
    }

    public void addNewSourceNode() {
        int count = sourceNodes.size() + 1;
        double y = railStart.getY() + (RUNG_SIZE * count);
        Point2D loc = new Point2D(railStart.getX(), y);
        SNode con = new SNode(loc);
        sourceNodes.add(con);
    }
}

package com.vts.jplceditor.ld.layout;

/**
 *
 * @author Vusumuzi_Tshabangu
 */
public class Size {
    private double width, height;

    public Size(double width, double height) {
        this.width = width;
        this.height = height;
    }

    public double getWidth() {
        return width;
    }

    public void setWidth(double width) {
        this.width = width;
    }

    public double getHeight() {
        return height;
    }

    public void setHeight(double height) {
        this.height = height;
    }
}

package com.vts.jplceditor.ld.instruction;

/**
 *
 * @author Vusumuzi_Tshabangu
 */
public class IR {

    private static final String core = "/com/vts/jplceditor/ld/instruction/";
    public static String Path_XIO = core + "relay/Xio.fxml";
```

```

public static String Path_XIC = core + "relay/Xic.fxml";
public static String Path_OTE = core + "relay/Ote.fxml";
public static String Path_OTL = core + "relay/Otl.fxml";
public static String Path_OTU = core + "relay/Otu.fxml";
public static String Path_TON = core + "timer/Ton.fxml";
public static String Path_CTU = core + "timer/Ctu.fxml";
public static String Path_CTD = core + "timer/Ctd.fxml";
public static String Path_EQU = core + "compare/Equ.fxml";
public static String Path_GRT = core + "compare/Grt.fxml";
public static String Path_LES = core + "compare/Les.fxml";
public static String Path_LIM = core + "compare/Lim.fxml";
public static String Path_MOV = core + "file/Mov.fxml";
public static String Path_MVM = core + "file/Mvm.fxml";
public static String Path_AND = core + "logical/And.fxml";
public static String Path_NOT = core + "logical/Not.fxml";
public static String Path_OR = core + "logical/Or.fxml";
public static String Path_XOR = core + "logical/Xor.fxml";
public static String Path_ADD = core + "math/Add.fxml";
public static String Path_DIV = core + "math/Div.fxml";
public static String Path_MUL = core + "math/Mul.fxml";
public static String Path_SUB = core + "math/Sub.fxml";
public static String DRAGGABLE_INSTRUCTION =
"/com/vts/jplceditor/ld/instruction/DraggableInstruction.fxml";
}

```

```

package com.vts.jplceditor.ld.instruction;

import com.vts.jplceditor.ld.instruction.compare.Equ;
import com.vts.jplceditor.ld.instruction.compare.Grt;
import com.vts.jplceditor.ld.instruction.compare.Les;
import com.vts.jplceditor.ld.instruction.compare.Lim;
import com.vts.jplceditor.ld.instruction.file.Mov;
import com.vts.jplceditor.ld.instruction.file.Mvm;
import com.vts.jplceditor.ld.instruction.logical.And;
import com.vts.jplceditor.ld.instruction.logical.Or;
import com.vts.jplceditor.ld.instruction.logical.Xor;
import com.vts.jplceditor.ld.instruction.math.Add;
import com.vts.jplceditor.ld.instruction.math.Div;
import com.vts.jplceditor.ld.instruction.math.Mul;
import com.vts.jplceditor.ld.instruction.math.Sub;
import com.vts.jplceditor.ld.instruction.relay.Xic;
import com.vts.jplceditor.ld.instruction.relay.Xio;
import com.vts.jplceditor.ld.instruction.relay.Ote;
import com.vts.jplceditor.ld.instruction.relay.Otl;
import com.vts.jplceditor.ld.instruction.relay.Otu;
import com.vts.jplceditor.ld.instruction.timer.Ctd;
import com.vts.jplceditor.ld.instruction.timer.Ctu;
import com.vts.jplceditor.ld.instruction.timer.Ton;

/**
 *
 * @author Vusivus
 */
public class InstructionUtil {

    public static Instruction getInstruction(String name) {

```

```

        switch (name) {
            case "XIC":
                return new Xic();
            case "XIO":
                return new Xio();
            case "OTE":
                return new Ote();
            case "OTL":
                return new Otl();
            case "OTU":
                return new Otu();
            case "TON":
                return new Ton();
            case "CTU":
                return new Ctu();
            case "CTD":
                return new Ctd();
            case "EQU":
                return new Equ();
            case "GRT":
                return new Grt();
            case "LES":
                return new Les();
            case "LIM":
                return new Lim();
            case "AND":
                return new And();
            case "OR":
                return new Or();
            case "XOR":
                return new Xor();
            case "ADD":
                return new Add();
            case "DIV":
                return new Div();
            case "MUL":
                return new Mul();
            case "SUB":
                return new Sub();
            case "MOV":
                return new Mov();
            case "MVM":
                return new Mvm();
            default:
                return null;
        }
    }

    public static boolean isBranch(String name){
        return "BRN".equals(name);
    }
}

package com.vts.jplceditor.ld.instruction;

import com.vts.jplceditor.compiler.plc.PLCInstruction;

```

```

import java.util.List;
import javafx.scene.layout.AnchorPane;

/**
 *
 * @author Vusivus
 */
public interface Instruction {
    public AnchorPane getInstance();
    public PLCInstruction getPLCInstruction();
}

package com.vts.jplceditor.ld.instruction;

/**
 *
 * @author Vusivus
 */
public class DragDataFormatter {

    public static String TagType_BIT = "bit";
    public static String TagType_BYTE = "byte";
    public static String TagType_INT = "integer";
    public static String TagType_FLOAT = "float";

    public static String DragData_TAG = "tag";
    public static String DragData_INSTRUCTION = "instruction";

    private static String[] tagType = new String[]{
        TagType_BIT, TagType_BYTE, TagType_INT, TagType_FLOAT};
    private static String[] dragDataType = new String[]{
        DragData_TAG, DragData_INSTRUCTION};

    public static String getTagType(String data) {
        if (validTagData(data)) {
            String[] split = splitData(data);
            return split[2];
        } else {
            return null;
        }
    }

    public static String getTagName(String data) {
        if (validTagData(data)) {
            String[] split = splitData(data);
            return split[1];
        } else {
            return null;
        }
    }

    public static String getInstructionName(String data) {
        if (validInstructionData(data)) {
            String[] split = splitData(data);
            return split[1];
        }
    }
}

```

```
        } else {
            return null;
        }
    }

    private static boolean validTagData(String data) {
        boolean valid;
        String[] split = splitData(data);
        if (split.length != 3) {
            return false;
        } else {
            valid = validateDataType(split[0]);
            if (!valid) {
                return false;
            } else {
                valid = validateTagType(split[2]);
                return valid;
            }
        }
    }

    private static boolean validInstructionData(String data) {
        boolean valid;
        String[] split = splitData(data);
        if (split.length != 3) {
            return false;
        } else {
            return split[0] == DragData_INSTRUCTION;
        }
    }

    private static String[] splitData(String data) {
        return data.split("_");
    }

    private static boolean validateDataType(String get) {
        return get.equals(DragData_TAG) || get.equals(DragData_INSTRUCTION);
    }

    private static boolean validateTagType(String get) {
        for (String tagType1 : tagType) {
            if (get.equals(tagType1)) {
                return true;
            }
        }
        return false;
    }

    public static String formatTag(String tagName, String tagType) {
        return DragData_TAG + "_" + tagName + "_" + tagType;
    }

    public static String formatInstruction(String iName, String iType) {
        return DragData_INSTRUCTION + "_" + iName + "_" + iType;
    }
}
```

```
package com.vts.jplceditor.ld.instruction.compare;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOperand;
import com.vts.jplceditor.compiler.plc.PLCOperator;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.Instruction;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.control.TextField;
import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundImage;
import javafx.scene.layout.CornerRadii;
import javafx.scene.paint.Color;

/**
 * FXML Controller class
 *
 * @author Vusman
 */
public class Equ extends AnchorPane implements Initializable, Instruction {

    @FXML
    private TextField in2TF;
    @FXML
    private TextField in1TF;
    private EventHandler<DragEvent> dragDropped;
    private EventHandler<DragEvent> dragOver;
    private EventHandler<DragEvent> dragEntered;
    private EventHandler<DragEvent> dragExited;
    private Point2D dragOffset;
    private boolean entered;
    private final List<PLCOperand> parameters;
    private final BackgroundImage fillDragOver;

    public Equ() {
        FXMLLoader fxmlLoader = new FXMLLoader(
            getClass().getResource(IR.Path_EQU));
        fxmlLoader.setRoot(this);
        fxmlLoader.setController(this);

        try {

```

```

        fxmlLoader.load();
    } catch (IOException exception) {

    }
parameters = new ArrayList<>();
parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
parameters.add(new PLCOperand(PLCDataType.INT, "In1"));
parameters.add(new PLCOperand(PLCDataType.INT, "In2"));
fillDragOver = new BackgroundFill(Color.BLUE, CornerRadii.EMPTY,
Insets.EMPTY);
}

/**
 * Initializes the controller class.
 */
@Override
public void initialize(URL url, ResourceBundle rb) {
    buildHandlers();
}

public void relocateTo(Point2D p) {
    Point2D point = getParent().sceneToLocal(p);
    relocate(point.getX(), point.getY());
}

private void buildHandlers() {

    dragOver = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                /* allow for moving */
                event.acceptTransferModes(TransferMode.COPY);
            }
            event.consume();
        }
    };

    dragDropped = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureTarget() != this) {
                /* data dropped */
/* if there is a string data on dragboard, read it and use it */
                Dragboard db = event.getDragboard();
                boolean success = false;
                if (db.hasString()) {

                    String data = db.getString();
                    String dataType = DragDataFormatter.getTagType(data);
                    if (validType(dataType)) {
                        String name = DragDataFormatter.getTagName(data);
                        if (event.getTarget().equals(in1TF)) {
                            in1TF.setText(name);
                        } else if (event.getTarget().equals(in2TF)) {

```

```

                in2TF.setText(name);
            }
        }
    }
    /* let the source know whether the string was
successfully
* transferred and used */
    event.setDropCompleted(success);

    event.consume();
}
};

dragEntered = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            if (event.getTarget().equals(in1TF)) {
                in1TF.setBackground(new Background(fillDragOver));
            }
            if (event.getTarget().equals(in2TF)) {
                in2TF.setBackground(new Background(fillDragOver));
            }
        }
        event.consume();
    }
};
dragExited = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            if (event.getTarget().equals(in1TF)) {
                in1TF.setBackground(Background.EMPTY);
            } else if (event.getTarget().equals(in2TF)) {
                in2TF.setBackground(Background.EMPTY);
            }
        }
        event.consume();
    }
};

in1TF.setOnDragOver(dragOver);
in1TF.setOnDragDropped(dragDropped);
in1TF.setOnDragEntered(dragEntered);
in1TF.setOnDragExited(dragExited);
in2TF.setOnDragOver(dragOver);
in2TF.setOnDragDropped(dragDropped);
in2TF.setOnDragEntered(dragEntered);
in2TF.setOnDragExited(dragExited);
}

@Override
public AnchorPane getInstance() {
    return this;
}

```

```

    }

    @Override
    public PLCInstruction getPLCInstruction() {
        parameters.get(2).setIdentity(in1TF.getText());
        parameters.get(3).setIdentity(in2TF.getText());
        PLCInstruction instruction = new PLCInstruction(PLCOperator.EQU);
        instruction.setOperands(parameters);
        return instruction;
    }

    private boolean validType(String type) {
        return type.equals(DragDataFormatter.TagType_INT) ||
        type.equals(DragDataFormatter.TagType_BYTE) ||
        type.equals(DragDataFormatter.TagType_FLOAT);
    }
}

package com.vts.jplceditor.ld.instruction.compare;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOperand;
import com.vts.jplceditor.compiler.plc.PLCOperator;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.Instruction;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Insets;
import javafx.geometry.Point2D;
import javafx.scene.control.TextField;
import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.CornerRadii;
import javafx.scene.paint.Color;

/**
 * FXML Controller class
 *
 * @author Vusman
 */
public class Grt extends AnchorPane implements Initializable, Instruction {

```

```

@FXML
private TextField in2TF;
@FXML
private TextField in1TF;
private EventHandler<DragEvent> dragDropped;
private EventHandler<DragEvent> dragOver;
private EventHandler<DragEvent> dragEntered;
private EventHandler<DragEvent> dragExited;
private Point2D dragOffset;
private boolean entered;
private final List<PLCOperand> parameters;
private final BackgroundFill fillDragOver;

public Grt() {
    FXMLLoader fxmlLoader = new FXMLLoader(
        getClass().getResource(IR.Path_GRT));
    fxmlLoader.setRoot(this);
    fxmlLoader.setController(this);

    try {
        fxmlLoader.load();
    } catch (IOException exception) {

    }
    parameters = new ArrayList<>();
    parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
    parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
    parameters.add(new PLCOperand(PLCDataType.INT, "In1"));
    parameters.add(new PLCOperand(PLCDataType.INT, "In2"));
    fillDragOver = new BackgroundFill(Color.BLUE, CornerRadii.EMPTY,
    Insets.EMPTY);
}

/**
 * Initializes the controller class.
 */
@Override
public void initialize(URL url, ResourceBundle rb) {
    buildHandlers();
}

public void relocateTo(Point2D p) {
    Point2D point = getParent().sceneToLocal(p);
    relocate(point.getX(), point.getY());
}

private void buildHandlers() {

    dragOver = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                /* allow for moving */
                event.acceptTransferModes(TransferMode.COPY);
            }
            event.consume();
        }
    };
}

```

```

        }

    };

    dragDropped = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureTarget() != this) {
                /* data dropped */
            /* if there is a string data on dragboard, read it and use it */
                Dragboard db = event.getDragboard();
                boolean success = false;
                if (db.hasString()) {

                    String data = db.getString();
                    String dataType = DragDataFormatter.getTagType(data);
                    if (validType(dataType)) {
                        String name = DragDataFormatter.getTagName(data);
                        if (event.getTarget().equals(in1TF)) {
                            in1TF.setText(name);
                        } else if (event.getTarget().equals(in2TF)) {
                            in2TF.setText(name);
                        }
                    }
                }
                /* let the source know whether the string was
                successfully
                * transferred and used */
                event.setDropCompleted(success);

                event.consume();
            }
        }
    };

    dragEntered = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                if (event.getTarget().equals(in1TF)) {
                    in1TF.setBackground(new Background(fillDragOver));
                }
                if (event.getTarget().equals(in2TF)) {
                    in2TF.setBackground(new Background(fillDragOver));
                }
            }
            event.consume();
        }
    };
    dragExited = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                if (event.getTarget().equals(in1TF)) {
                    in1TF.setBackground(Background.EMPTY);
                } else if (event.getTarget().equals(in2TF)) {

```

```

                in2TF.setBackground(Background.EMPTY);
            }
        }
        event.consume();
    }
};

in1TF.setOnDragOver(dragOver);
in1TF.setOnDragDropped(dragDropped);
in1TF.setOnDragEntered(dragEntered);
in1TF.setOnDragExited(dragExited);
in2TF.setOnDragOver(dragOver);
in2TF.setOnDragDropped(dragDropped);
in2TF.setOnDragEntered(dragEntered);
in2TF.setOnDragExited(dragExited);
}

@Override
public AnchorPane getInstance() {
    return this;
}

@Override
public PLCInstruction getPLCInstruction() {
    parameters.get(2).setIdentity(in1TF.getText());
    parameters.get(3).setIdentity(in2TF.getText());
    PLCInstruction instruction = new PLCInstruction(PLCOperator.GRT);
    instruction.setOperands(parameters);
    return instruction;
}

private boolean validType(String type) {
    return type.equals(DragDataFormatter.TagType_INT) ||
    type.equals(DragDataFormatter.TagType_BYTE) ||
    type.equals(DragDataFormatter.TagType_FLOAT);
}
}

package com.vts.jplceditor.ld.instruction.compare;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.compiler.plc.PLCOoperator;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.Instruction;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
```

```

import javafx.geometry.Insets;
import javafx.geometry.Point2D;
import javafx.scene.control.TextField;
import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.CornerRadii;
import javafx.scene.paint.Color;

/**
 * FXML Controller class
 *
 * @author Vusman
 */
public class Les extends AnchorPane implements Initializable, Instruction {

    @FXML
    private TextField in2TF;
    @FXML
    private TextField in1TF;
    private EventHandler<DragEvent> dragDropped;
    private EventHandler<DragEvent> dragOver;
    private EventHandler<DragEvent> dragEntered;
    private EventHandler<DragEvent> dragExited;
    private Point2D dragOffset;
    private boolean entered;
    private final List<PLCOperand> parameters;
    private final BackgroundFill fillDragOver;

    public Les() {
        FXMLLoader fxmlLoader = new FXMLLoader(
            getClass().getResource(IR.Path_LES));
        fxmlLoader.setRoot(this);
        fxmlLoader.setController(this);

        try {
            fxmlLoader.load();
        } catch (IOException exception) {

        }
        parameters = new ArrayList<>();
        parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
        parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
        parameters.add(new PLCOperand(PLCDataType.INT, "In1"));
        parameters.add(new PLCOperand(PLCDataType.INT, "In2"));
        fillDragOver = new BackgroundFill(Color.BLUE, CornerRadii.EMPTY,
        Insets.EMPTY);
    }

    /**
     * Initializes the controller class.
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {

```

```

        buildHandlers();
    }

    public void relocateTo(Point2D p) {
        Point2D point = getParent().sceneToLocal(p);
        relocate(point.getX(), point.getY());
    }

    private void buildHandlers() {

        dragOver = new EventHandler<DragEvent>() {
            @Override
            public void handle(DragEvent event) {
                if (event.getGestureSource() != this
                    && event.getDragboard().hasString()) {
                    /* allow for moving */
                    event.acceptTransferModes(TransferMode.COPY);
                }
                event.consume();
            }
        };

        dragDropped = new EventHandler<DragEvent>() {
            @Override
            public void handle(DragEvent event) {
                if (event.getGestureTarget() != this) {
                    /* data dropped */
                    /* if there is a string data on dragboard, read it and use it */
                    Dragboard db = event.getDragboard();
                    boolean success = false;
                    if (db.hasString()) {

                        String data = db.getString();
                        String dataType = DragDataFormatter.getTagType(data);
                        if (dataType.equals(DragDataFormatter.TagType_INT)) {
                            String name = DragDataFormatter.getTagName(data);
                            if (event.getTarget().equals(in1TF)) {
                                in1TF.setText(name);
                            } else if (event.getTarget().equals(in2TF)) {
                                in2TF.setText(name);
                            }
                        }
                    }
                    /* let the source know whether the string was
                     * successfully
                     * transferred and used */
                    event.setDropCompleted(success);

                    event.consume();
                }
            }
        };

        dragEntered = new EventHandler<DragEvent>() {
            @Override
            public void handle(DragEvent event) {
                if (event.getGestureSource() != this

```

```

        && event.getDragboard().hasString() ) {
    if (event.getTarget().equals(in1TF)) {
        in1TF.setBackground(new Background(fillDragOver));
    }
    if (event.getTarget().equals(in2TF)) {
        in2TF.setBackground(new Background(fillDragOver));
    }
}
event.consume();
}
};

dragExited = new EventHandler<DragEvent>() {
@Override
public void handle(DragEvent event) {
    if (event.getGestureSource() != this
        && event.getDragboard().hasString() ) {
        if (event.getTarget().equals(in1TF)) {
            in1TF.setBackground(Background.EMPTY);
        } else if (event.getTarget().equals(in2TF)) {
            in2TF.setBackground(Background.EMPTY);
        }
    }
    event.consume();
}
};

in1TF.setOnDragOver(dragOver);
in1TF.setOnDragDropped(dragDropped);
in1TF.setOnDragEntered(dragEntered);
in1TF.setOnDragExited(dragExited);
in2TF.setOnDragOver(dragOver);
in2TF.setOnDragDropped(dragDropped);
in2TF.setOnDragEntered(dragEntered);
in2TF.setOnDragExited(dragExited);
}

@Override
public AnchorPane getInstance() {
    return this;
}

@Override
public PLCInstruction getPLCInstruction() {
    parameters.get(2).setIdentity(in1TF.getText());
    parameters.get(3).setIdentity(in2TF.getText());
    PLCInstruction instruction = new PLCInstruction(PLCOperator.LES);
    instruction.setOperands(parameters);
    return instruction;
}

private boolean validType(String type){
    return type.equals(DragDataFormatter.TagType_INT) ||
type.equals(DragDataFormatter.TagType_BYTE) ||
type.equals(DragDataFormatter.TagType_FLOAT);
}

}

```

```
package com.vts.jplceditor.ld.instruction.compare;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOperand;
import com.vts.jplceditor.compiler.plc.PLCOperator;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.Instruction;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.control.TextField;
import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.CornerRadii;
import javafx.scene.paint.Color;

/**
 * FXML Controller class
 *
 * @author Vusman
 */
public class Lim extends AnchorPane implements Initializable, Instruction {

    @FXML
    private TextField lowTF;
    @FXML
    private TextField testTF;
    @FXML
    private TextField highTF;
    private EventHandler<DragEvent> dragDropped;
    private EventHandler<DragEvent> dragOver;
    private EventHandler<DragEvent> dragEntered;
    private EventHandler<DragEvent> dragExited;
    private Point2D dragOffset;
    private boolean entered;
    private final List<PLCOperand> parameters;
    private final BackgroundFill fillDragOver;

    public Lim() {
        FXMLLoader fxmlLoader = new FXMLLoader(
            getClass().getResource(IR.Path_LIM));
        fxmlLoader.setRoot(this);
        fxmlLoader.setController(this);
    }
}
```

```

try {
    fxmlLoader.load();
} catch (IOException exception) {

}

parameters = new ArrayList<>();
parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
parameters.add(new PLCOperand(PLCDataType.INT, "low"));
parameters.add(new PLCOperand(PLCDataType.INT, "test"));
parameters.add(new PLCOperand(PLCDataType.INT, "high"));
fillDragOver = new BackgroundFill(Color.BLUE, CornerRadii.EMPTY,
Insets.EMPTY);
}

/**
 * Initializes the controller class.
 */
@Override
public void initialize(URL url, ResourceBundle rb) {
    buildHandlers();
}

public void relocateTo(Point2D p) {
    Point2D point = getParent().sceneToLocal(p);
    relocate(point.getX(), point.getY());
}

private void buildHandlers() {

    dragOver = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                /* allow for moving */
                event.acceptTransferModes(TransferMode.COPY);
            }
            event.consume();
        }
    };

    dragDropped = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureTarget() != this) {
                /* data dropped */
            }
        }
    };
}

/* if there is a string data on dragboard, read it and use it */
Dragboard db = event.getDragboard();
boolean success = false;
if (db.hasString()) {

    String data = db.getString();
    String dataType = DragDataFormatter.getTagType(data);
    if (validType(dataType)) {
        String name = DragDataFormatter.getTagName(data);
    }
}

```

```

        if (event.getTarget().equals(lowTF)) {
            lowTF.setText(name);
        }
        if (event.getTarget().equals(testTF)) {
            testTF.setText(name);
        }
        if (event.getTarget().equals(highTF)) {
            highTF.setText(name);
        }
    }
}
/* let the source know whether the string was
successfully
* transferred and used */
event.setDropCompleted(success);

event.consume();
}
}
};

dragEntered = new EventHandler<DragEvent>() {
@Override
public void handle(DragEvent event) {
    if (event.getGestureSource() != this
        && event.getDragboard().hasString()) {
        if (event.getTarget().equals(lowTF)) {
            lowTF.setBackground(new Background(fillDragOver));
        } else if (event.getTarget().equals(testTF)) {
            testTF.setBackground(new Background(fillDragOver));
        } else if (event.getTarget().equals(highTF)) {
            highTF.setBackground(new Background(fillDragOver));
        }
    }
    event.consume();
}
};

dragExited = new EventHandler<DragEvent>() {
@Override
public void handle(DragEvent event) {
    if (event.getGestureSource() != this
        && event.getDragboard().hasString()) {
        if (event.getTarget().equals(lowTF)) {
            lowTF.setBackground(Background.EMPTY);
        } else if (event.getTarget().equals(testTF)) {
            testTF.setBackground(Background.EMPTY);
        } else if (event.getTarget().equals(highTF)) {
            highTF.setBackground(Background.EMPTY);
        }
    }
    event.consume();
}
};

lowTF.setOnDragOver(dragOver);
lowTF.setOnDragDropped(dragDropped);

```

```
        lowTF.setOnDragEntered(dragEntered);
        lowTF.setOnDragExited(dragExited);
        testTF.setOnDragOver(dragOver);
        testTF.setOnDragDropped(dragDropped);
        testTF.setOnDragEntered(dragEntered);
        testTF.setOnDragExited(dragExited);
        highTF.setOnDragOver(dragOver);
        highTF.setOnDragDropped(dragDropped);
        highTF.setOnDragEntered(dragEntered);
        highTF.setOnDragExited(dragExited);
    }

    @Override
    public AnchorPane getInstance() {
        return this;
    }

    @Override
    public PLCInstruction getPLCInstruction() {
        parameters.get(2).setIdentity(lowTF.getText());
        parameters.get(3).setIdentity(testTF.getText());
        parameters.get(4).setIdentity(highTF.getText());
        PLCInstruction instruction = new PLCInstruction(PLCOperator.LIM);
        instruction.setOperands(parameters);
        return instruction;
    }

    private boolean validType(String type) {
        return type.equals(DragDataFormatter.TagType_INT) ||
               type.equals(DragDataFormatter.TagType_BYTE) ||
               type.equals(DragDataFormatter.TagType_FLOAT);
    }
}
```

```
package com.vts.jplceditor.ld.instruction.file;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.compiler.plc.PLCOOperator;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.Instruction;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Insets;
import javafx.geometry.Point2D;
import javafx.scene.control.TextField;
```

```

import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.CornerRadii;
import javafx.scene.paint.Color;

/**
 * FXML Controller class
 *
 * @author Vusman
 */
public class Mov extends AnchorPane implements Initializable, Instruction {

    @FXML
    private TextField destTF;
    @FXML
    private TextField sourceTF;
    private EventHandler<DragEvent> dragDropped;
    private EventHandler<DragEvent> dragOver;
    private EventHandler<DragEvent> dragEntered;
    private EventHandler<DragEvent> dragExited;
    private Point2D dragOffset;
    private boolean entered;
    private final List<PLCOperand> parameters;
    private final BackgroundFill fillDragOver;

    public Mov() {
        FXMLLoader fxmlLoader = new FXMLLoader(
            getClass().getResource(IR.Path_MOV));
        fxmlLoader.setRoot(this);
        fxmlLoader.setController(this);

        try {
            fxmlLoader.load();
        } catch (IOException exception) {

        }
        parameters = new ArrayList<>();
        parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
        parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
        parameters.add(new PLCOperand(PLCDataType.INT, "Source"));
        parameters.add(new PLCOperand(PLCDataType.INT, "Destination"));
        fillDragOver = new BackgroundFill(Color.BLUE, CornerRadii.EMPTY,
        Insets.EMPTY);
    }

    /**
     * Initializes the controller class.
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        buildHandlers();
    }
}

```

```

public void relocateTo(Point2D p) {
    Point2D point = getParent().sceneToLocal(p);
    relocate(point.getX(), point.getY());
}

private void buildHandlers() {

    dragOver = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                /* allow for moving */
                event.acceptTransferModes(TransferMode.COPY);
            }
            event.consume();
        }
    };

    dragDropped = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureTarget() != this) {
                /* data dropped */
                /* if there is a string data on dragboard, read it and use it */
                Dragboard db = event.getDragboard();
                boolean success = false;
                if (db.hasString()) {

                    String data = db.getString();
                    String dataType = DragDataFormatter.getTagType(data);
                    if (validType(dataType)) {
                        String name = DragDataFormatter.getTagName(data);
                        if (event.getTarget().equals(sourceTF)) {
                            sourceTF.setText(name);
                        } else if (event.getTarget().equals(destTF)) {
                            destTF.setText(name);
                        }
                    }
                }
                /* let the source know whether the string was
                successfully
                * transferred and used */
                event.setDropCompleted(success);

                event.consume();
            }
        }
    };

    dragEntered = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                if (event.getTarget().equals(sourceTF)) {
                    sourceTF.setBackground(new Background(fillDragOver));
                }
            }
        }
    };
}

```

```

        }
        if (event.getTarget().equals(destTF)) {
            destTF.setBackground(new Background(fillDragOver));
        }
    }
    event.consume();
}
};

dragExited = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            if (event.getTarget().equals(sourceTF)) {
                sourceTF.setBackground(Background.EMPTY);
            } else if (event.getTarget().equals(destTF)) {
                destTF.setBackground(Background.EMPTY);
            }
        }
        event.consume();
    }
};

sourceTF.setOnDragOver(dragOver);
sourceTF.setOnDragDropped(dragDropped);
sourceTF.setOnDragEntered(dragEntered);
sourceTF.setOnDragExited(dragExited);
destTF.setOnDragOver(dragOver);
destTF.setOnDragDropped(dragDropped);
destTF.setOnDragEntered(dragEntered);
destTF.setOnDragExited(dragExited);
}

@Override
public AnchorPane getInstance() {
    return this;
}

@Override
public PLCInstruction getPLCInstruction() {
    parameters.get(2).setIdentity(sourceTF.getText());
    parameters.get(3).setIdentity(destTF.getText());
    PLCInstruction instruction = new PLCInstruction(PLCOperator.MOV);
    instruction.setOperands(parameters);
    return instruction;
}

private boolean validType(String type) {
    return type.equals(DragDataFormatter.TagType_INT) ||
    type.equals(DragDataFormatter.TagType_BYTE) ||
    type.equals(DragDataFormatter.TagType_FLOAT);
}

}

```

```
package com.vts.jplceditor.ld.instruction.file;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOperand;
import com.vts.jplceditor.compiler.plc.PLCOperator;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.Instruction;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.control.TextField;
import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundImage;
import javafx.scene.layout.CornerRadii;
import javafx.scene.paint.Color;

/**
 * FXML Controller class
 *
 * @author Vusman
 */
public class Mvm extends AnchorPane implements Initializable, Instruction {

    @FXML
    private TextField sourceTF;
    @FXML
    private TextField maskTF;
    @FXML
    private TextField destTF;
    private EventHandler<DragEvent> dragDropped;
    private EventHandler<DragEvent> dragOver;
    private EventHandler<DragEvent> dragEntered;
    private EventHandler<DragEvent> dragExited;
    private Point2D dragOffset;
    private boolean entered;
    private final List<PLCOperand> parameters;
    private final BackgroundImage fillDragOver;

    public Mvm() {
        FXMLLoader fxmlLoader = new FXMLLoader(
            getClass().getResource(IR.Path_MVM));
        fxmlLoader.setRoot(this);
        fxmlLoader.setController(this);
    }
}
```

```

try {
    fxmlLoader.load();
} catch (IOException exception) {

}

parameters = new ArrayList<>();
parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
parameters.add(new PLCOperand(PLCDataType.INT, "low"));
parameters.add(new PLCOperand(PLCDataType.INT, "test"));
parameters.add(new PLCOperand(PLCDataType.INT, "high"));
fillDragOver = new BackgroundFill(Color.BLUE, CornerRadii.EMPTY,
Insets.EMPTY);
}

/**
 * Initializes the controller class.
 */
@Override
public void initialize(URL url, ResourceBundle rb) {
    buildHandlers();
}

public void relocateTo(Point2D p) {
    Point2D point = getParent().sceneToLocal(p);
    relocate(point.getX(), point.getY());
}

private void buildHandlers() {

    dragOver = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                /* allow for moving */
                event.acceptTransferModes(TransferMode.COPY);
            }
            event.consume();
        }
    };

    dragDropped = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureTarget() != this) {
                /* data dropped */
            }
        }
    };
}

/* if there is a string data on dragboard, read it and use it */
Dragboard db = event.getDragboard();
boolean success = false;
if (db.hasString()) {

    String data = db.getString();
    String dataType = DragDataFormatter.getTagType(data);
    if (validType(dataType)) {
        String name = DragDataFormatter.getTagName(data);
    }
}

```

```

        if (event.getTarget().equals(sourceTF)) {
            sourceTF.setText(name);
        }
        if (event.getTarget().equals(maskTF)) {
            maskTF.setText(name);
        }
        if (event.getTarget().equals(destTF)) {
            destTF.setText(name);
        }
    }
}
/* let the source know whether the string was
successfully
* transferred and used */
event.setDropCompleted(success);

event.consume();
}
}
};

dragEntered = new EventHandler<DragEvent>() {
@Override
public void handle(DragEvent event) {
    if (event.getGestureSource() != this
        && event.getDragboard().hasString()) {
        if (event.getTarget().equals(sourceTF)) {
            sourceTF.setBackground(new Background(fillDragOver));
        } else if (event.getTarget().equals(maskTF)) {
            maskTF.setBackground(new Background(fillDragOver));
        } else if (event.getTarget().equals(destTF)) {
            destTF.setBackground(new Background(fillDragOver));
        }
    }
    event.consume();
}
};

dragExited = new EventHandler<DragEvent>() {
@Override
public void handle(DragEvent event) {
    if (event.getGestureSource() != this
        && event.getDragboard().hasString()) {
        if (event.getTarget().equals(sourceTF)) {
            sourceTF.setBackground(Background.EMPTY);
        } else if (event.getTarget().equals(maskTF)) {
            maskTF.setBackground(Background.EMPTY);
        } else if (event.getTarget().equals(destTF)) {
            destTF.setBackground(Background.EMPTY);
        }
    }
    event.consume();
}
};

sourceTF.setOnDragOver(dragOver);
sourceTF.setOnDragDropped(dragDropped);

```

```
sourceTF.setOnDragEntered(dragEntered);
sourceTF.setOnDragExited(dragExited);
maskTF.setOnDragOver(dragOver);
maskTF.setOnDragDropped(dragDropped);
maskTF.setOnDragEntered(dragEntered);
maskTF.setOnDragExited(dragExited);
destTF.setOnDragOver(dragOver);
destTF.setOnDragDropped(dragDropped);
destTF.setOnDragEntered(dragEntered);
destTF.setOnDragExited(dragExited);
}

@Override
public AnchorPane getInstance() {
    return this;
}
@Override
public PLCInstruction getPLCInstruction() {
    parameters.get(2).setIdentity(sourceTF.getText());
    parameters.get(3).setIdentity(maskTF.getText());
    parameters.get(4).setIdentity(destTF.getText());
    PLCInstruction instruction = new PLCInstruction(PLCOperator.MVM);
    instruction.setOperands(parameters);
    return instruction;
}

private boolean validType(String type) {
    return type.equals(DragDataFormatter.TagType_INT) ||
    type.equals(DragDataFormatter.TagType_BYTE) ||
    type.equals(DragDataFormatter.TagType_FLOAT);
}
```

```
package com.vts.jplceditor.ld.instruction.logical;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.compiler.plc.PLCOOperator;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.Instruction;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Insets;
import javafx.geometry.Point2D;
import javafx.scene.control.TextField;
```

```

import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.CornerRadii;
import javafx.scene.paint.Color;

/**
 * FXML Controller class
 *
 * @author Vusman
 */
public class And extends AnchorPane implements Initializable, Instruction {

    @FXML
    private TextField in2TF;
    @FXML
    private TextField in1TF;
    @FXML
    private TextField destTF;
    private EventHandler<DragEvent> dragDropped;
    private EventHandler<DragEvent> dragOver;
    private EventHandler<DragEvent> dragEntered;
    private EventHandler<DragEvent> dragExited;
    private Point2D dragOffset;
    private boolean entered;
    private final List<PLCOperand> parameters;
    private final BackgroundFill fillDragOver;

    public And() {
        FXMLLoader fxmlLoader = new FXMLLoader(
            getClass().getResource(IR.Path_AND));
        fxmlLoader.setRoot(this);
        fxmlLoader.setController(this);

        try {
            fxmlLoader.load();
        } catch (IOException exception) {

        }
        parameters = new ArrayList<>();
        parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
        parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
        parameters.add(new PLCOperand(PLCDataType.INT, "In1"));
        parameters.add(new PLCOperand(PLCDataType.INT, "In2"));
        parameters.add(new PLCOperand(PLCDataType.INT, "Dest"));
        fillDragOver = new BackgroundFill(Color.BLUE, CornerRadii.EMPTY,
        Insets.EMPTY);
    }

    /**
     * Initializes the controller class.
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {

```

```

        buildHandlers();
    }

    public void relocateTo(Point2D p) {
        Point2D point = getParent().sceneToLocal(p);
        relocate(point.getX(), point.getY());
    }

    private void buildHandlers() {

        dragOver = new EventHandler<DragEvent>() {
            @Override
            public void handle(DragEvent event) {
                if (event.getGestureSource() != this
                    && event.getDragboard().hasString()) {
                    /* allow for moving */
                    event.acceptTransferModes(TransferMode.COPY);
                }
                event.consume();
            }
        };
    }

    dragDropped = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureTarget() != this) {
                /* data dropped */
                /* if there is a string data on dragboard, read it and use it */
                Dragboard db = event.getDragboard();
                boolean success = false;
                if (db.hasString()) {

                    String data = db.getString();
                    String dataType = DragDataFormatter.getTagType(data);
                    if (validType(dataType)) {
                        String name = DragDataFormatter.getTagName(data);
                        if (event.getTarget().equals(in1TF)) {
                            in1TF.setText(name);
                        } else if (event.getTarget().equals(in2TF)) {
                            in2TF.setText(name);
                        } else if (event.getTarget().equals(destTF)) {
                            destTF.setText(name);
                        }
                    }
                }
                /* let the source know whether the string was
                successfully
                * transferred and used */
                event.setDropCompleted(success);

                event.consume();
            }
        }
    };

    dragEntered = new EventHandler<DragEvent>() {
        @Override

```

```

        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                if (event.getTarget().equals(in1TF)) {
                    in1TF.setBackground(new Background(fillDragOver));
                }
                if (event.getTarget().equals(in2TF)) {
                    in2TF.setBackground(new Background(fillDragOver));
                }
                if (event.getTarget().equals(destTF)) {
                    destTF.setBackground(new Background(fillDragOver));
                }
            }
            event.consume();
        }
    };
    dragExited = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                if (event.getTarget().equals(in1TF)) {
                    in1TF.setBackground(Background.EMPTY);
                } else if (event.getTarget().equals(in2TF)) {
                    in2TF.setBackground(Background.EMPTY);
                } else if (event.getTarget().equals(destTF)) {
                    destTF.setBackground(Background.EMPTY);
                }
            }
            event.consume();
        }
    };
    in1TF.setOnDragOver(dragOver);
    in1TF.setOnDragDropped(dragDropped);
    in1TF.setOnDragEntered(dragEntered);
    in1TF.setOnDragExited(dragExited);
    in2TF.setOnDragOver(dragOver);
    in2TF.setOnDragDropped(dragDropped);
    in2TF.setOnDragEntered(dragEntered);
    in2TF.setOnDragExited(dragExited);
    destTF.setOnDragOver(dragOver);
    destTF.setOnDragDropped(dragDropped);
    destTF.setOnDragEntered(dragEntered);
    destTF.setOnDragExited(dragExited);
}
}

@Override
public AnchorPane getInstance() {
    return this;
}

@Override
public PLCInstruction getPLCInstruction() {
    parameters.get(2).setIdentity(in1TF.getText());
    parameters.get(3).setIdentity(in2TF.getText());
    parameters.get(4).setIdentity(destTF.getText());
}

```

```
    PLCInstruction instruction = new PLCInstruction(PLCOperator.AND);
    instruction.setOperands(parameters);
    return instruction;
}

private boolean validType(String type) {
    return type.equals(DragDataFormatter.TagType_INT) ||
    type.equals(DragDataFormatter.TagType_BYTE) ||
    type.equals(DragDataFormatter.TagType_FLOAT);
}

}

package com.vts.jplceditor.ld.instruction.logical;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.compiler.plc.PLCOperator;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.Instruction;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Insets;
import javafx.geometry.Point2D;
import javafx.scene.control.TextField;
import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.CornerRadii;
import javafx.scene.paint.Color;

/**
 * FXML Controller class
 *
 * @author Vusman
 */
public class Or extends AnchorPane implements Initializable, Instruction {

    @FXML
    private TextField in2TF;
    @FXML
    private TextField in1TF;
```

```

@FXML
private TextField destTF;
private EventHandler<DragEvent> dragDropped;
private EventHandler<DragEvent> dragOver;
private EventHandler<DragEvent> dragEntered;
private EventHandler<DragEvent> dragExited;
private Point2D dragOffset;
private boolean entered;
private final List<PLCOperand> parameters;
private final BackgroundFill fillDragOver;

public Or() {
    FXMLLoader fxmlLoader = new FXMLLoader(
        getClass().getResource(IR.Path_OR));
    fxmlLoader.setRoot(this);
    fxmlLoader.setController(this);

    try {
        fxmlLoader.load();
    } catch (IOException exception) {

    }
    parameters = new ArrayList<>();
    parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
    parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
    parameters.add(new PLCOperand(PLCDataType.INT, "In1"));
    parameters.add(new PLCOperand(PLCDataType.INT, "In2"));
    parameters.add(new PLCOperand(PLCDataType.INT, "Dest"));
    fillDragOver = new BackgroundFill(Color.BLUE, CornerRadii.EMPTY,
    Insets.EMPTY);
}

/**
 * Initializes the controller class.
 */
@Override
public void initialize(URL url, ResourceBundle rb) {
    buildHandlers();
}

public void relocateTo(Point2D p) {
    Point2D point = getParent().sceneToLocal(p);
    relocate(point.getX(), point.getY());
}

private void buildHandlers() {

    dragOver = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                /* allow for moving */
                event.acceptTransferModes(TransferMode.COPY);
            }
            event.consume();
        }
    };
}

```

```

};

dragDropped = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureTarget() != this) {
            /* data dropped */
        /* if there is a string data on dragboard, read it and use it */
            Dragboard db = event.getDragboard();
            boolean success = false;
            if (db.hasString()) {

                String data = db.getString();
                String dataType = DragDataFormatter.getTagType(data);
                if (validType(dataType)) {
                    String name = DragDataFormatter.getTagName(data);
                    if (event.getTarget().equals(in1TF)) {
                        in1TF.setText(name);
                    } else if (event.getTarget().equals(in2TF)) {
                        in2TF.setText(name);
                    } else if (event.getTarget().equals(destTF)) {
                        destTF.setText(name);
                    }
                }
            }
            /* let the source know whether the string was
            successfully
            * transferred and used */
            event.setDropCompleted(success);

            event.consume();
        }
    }
};

dragEntered = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            if (event.getTarget().equals(in1TF)) {
                in1TF.setBackground(new Background(fillDragOver));
            }
            if (event.getTarget().equals(in2TF)) {
                in2TF.setBackground(new Background(fillDragOver));
            }
            if (event.getTarget().equals(destTF)) {
                destTF.setBackground(new Background(fillDragOver));
            }
        }
        event.consume();
    }
};

dragExited = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this

```

```

        && event.getDragboard().hasString()) {
    if (event.getTarget().equals(in1TF)) {
        in1TF.setBackground(Background.EMPTY);
    } else if (event.getTarget().equals(in2TF)) {
        in2TF.setBackground(Background.EMPTY);
    } else if (event.getTarget().equals(destTF)) {
        destTF.setBackground(Background.EMPTY);
    }
}
event.consume();
};

in1TF.setOnDragOver(dragOver);
in1TF.setOnDragDropped(dragDropped);
in1TF.setOnDragEntered(dragEntered);
in1TF.setOnDragExited(dragExited);
in2TF.setOnDragOver(dragOver);
in2TF.setOnDragDropped(dragDropped);
in2TF.setOnDragEntered(dragEntered);
in2TF.setOnDragExited(dragExited);
destTF.setOnDragOver(dragOver);
destTF.setOnDragDropped(dragDropped);
destTF.setOnDragEntered(dragEntered);
destTF.setOnDragExited(dragExited);
}

@Override
public AnchorPane getInstance() {
    return this;
}

@Override
public PLCInstruction getPLCInstruction() {
    parameters.get(2).setIdentity(in1TF.getText());
    parameters.get(3).setIdentity(in2TF.getText());
    parameters.get(4).setIdentity(destTF.getText());
    PLCInstruction instruction = new PLCInstruction(PLCOperator.OR);
    instruction.setOperands(parameters);
    return instruction;
}

private boolean validType(String type) {
    return type.equals(DragDataFormatter.TagType_INT) ||
    type.equals(DragDataFormatter.TagType_BYTE) ||
    type.equals(DragDataFormatter.TagType_FLOAT);
}

}

package com.vts.jplceditor.ld.instruction.logical;

import com.vts.jplceditor.compiler.plc.PLCDatatype;

```

```
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.compiler.plc.PLCOOperator;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.Instruction;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Insets;
import javafx.geometry.Point2D;
import javafx.scene.control.TextField;
import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.CornerRadii;
import javafx.scene.paint.Color;

/**
 * FXML Controller class
 *
 * @author Vusman
 */
public class Xor extends AnchorPane implements Initializable, Instruction {

    @FXML
    private TextField in2TF;
    @FXML
    private TextField in1TF;
    @FXML
    private TextField destTF;
    private EventHandler<DragEvent> dragDropped;
    private EventHandler<DragEvent> dragOver;
    private EventHandler<DragEvent> dragEntered;
    private EventHandler<DragEvent> dragExited;
    private Point2D dragOffset;
    private boolean entered;
    private final List<PLCOOperand> parameters;
    private final BackgroundFill fillDragOver;

    public Xor() {
        FXMLLoader fxmlLoader = new FXMLLoader(
            getClass().getResource(IR.Path_XOR));
        fxmlLoader.setRoot(this);
        fxmlLoader.setController(this);

        try {
            fxmlLoader.load();
        
```

```

    } catch (IOException exception) {
    }
parameters = new ArrayList<>();
parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
parameters.add(new PLCOperand(PLCDataType.INT, "In1"));
parameters.add(new PLCOperand(PLCDataType.INT, "In2"));
parameters.add(new PLCOperand(PLCDataType.INT, "Dest"));
fillDragOver = new BackgroundFill(Color.BLUE, CornerRadii.EMPTY,
Insets.EMPTY);
}

/**
 * Initializes the controller class.
 */
@Override
public void initialize(URL url, ResourceBundle rb) {
    buildHandlers();
}

public void relocateTo(Point2D p) {
    Point2D point = getParent().sceneToLocal(p);
    relocate(point.getX(), point.getY());
}

private void buildHandlers() {

    dragOver = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                /* allow for moving */
                event.acceptTransferModes(TransferMode.COPY);
            }
            event.consume();
        }
    };
}

dragDropped = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureTarget() != this) {
            /* data dropped */
/* if there is a string data on dragboard, read it and use it */
            Dragboard db = event.getDragboard();
            boolean success = false;
            if (db.hasString()) {

                String data = db.getString();
                String dataType = DragDataFormatter.getTagType(data);
                if (validType(dataType)) {
                    String name = DragDataFormatter.getTagName(data);
                    if (event.getTarget().equals(in1TF)) {
                        in1TF.setText(name);
                    } else if (event.getTarget().equals(in2TF)) {

```

```

                in2TF.setText(name);
            } else if (event.getTarget().equals(destTF)) {
                destTF.setText(name);
            }
        }
    }
    /* let the source know whether the string was
successfully
* transferred and used */
    event.setDropCompleted(success);

    event.consume();
}
};

dragEntered = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            if (event.getTarget().equals(in1TF)) {
                in1TF.setBackground(new Background(fillDragOver));
            }
            if (event.getTarget().equals(in2TF)) {
                in2TF.setBackground(new Background(fillDragOver));
            }
            if (event.getTarget().equals(destTF)) {
                destTF.setBackground(new Background(fillDragOver));
            }
        }
        event.consume();
    }
};

dragExited = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            if (event.getTarget().equals(in1TF)) {
                in1TF.setBackground(Background.EMPTY);
            } else if (event.getTarget().equals(in2TF)) {
                in2TF.setBackground(Background.EMPTY);
            } else if (event.getTarget().equals(destTF)) {
                destTF.setBackground(Background.EMPTY);
            }
        }
        event.consume();
    }
};

in1TF.setOnDragOver(dragOver);
in1TF.setOnDragDropped(dragDropped);
in1TF.setOnDragEntered(dragEntered);
in1TF.setOnDragExited(dragExited);
in2TF.setOnDragOver(dragOver);
in2TF.setOnDragDropped(dragDropped);

```

```
        in2TF.setOnDragEntered(dragEntered);
        in2TF.setOnDragExited(dragExited);
        destTF.setOnDragOver(dragOver);
        destTF.setOnDragDropped(dragDropped);
        destTF.setOnDragEntered(dragEntered);
        destTF.setOnDragExited(dragExited);
    }

@Override
public AnchorPane getInstance() {
    return this;
}

@Override
public PLCInstruction getPLCInstruction() {
    parameters.get(2).setIdentity(in1TF.getText());
    parameters.get(3).setIdentity(in2TF.getText());
    parameters.get(4).setIdentity(destTF.getText());
    PLCInstruction instruction = new PLCInstruction(PLCOperator.XOR);
    instruction.setOperands(parameters);
    return instruction;
}

private boolean validType(String type) {
    return type.equals(DragDataFormatter.TagType_INT) ||
    type.equals(DragDataFormatter.TagType_BYTE) ||
    type.equals(DragDataFormatter.TagType_FLOAT);
}

}

package com.vts.jplceditor.ld.instruction.math;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.compiler.plc.PLCOperator;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.Instruction;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Insets;
import javafx.geometry.Point2D;
import javafx.scene.control.TextField;
import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
```

```

import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.CornerRadii;
import javafx.scene.paint.Color;

/**
 * FXML Controller class
 *
 * @author Vusman
 */
public class Add extends AnchorPane implements Initializable, Instruction {

    @FXML
    private TextField in2TF;
    @FXML
    private TextField in1TF;
    @FXML
    private TextField destTF;
    private EventHandler<DragEvent> dragDropped;
    private EventHandler<DragEvent> dragOver;
    private EventHandler<DragEvent> dragEntered;
    private EventHandler<DragEvent> dragExited;
    private Point2D dragOffset;
    private boolean entered;
    private final List<PLCOperand> parameters;
    private final BackgroundFill fillDragOver;

    public Add() {
        FXMLLoader fxmlLoader = new FXMLLoader(
            getClass().getResource(IR.Path_ADD));
        fxmlLoader.setRoot(this);
        fxmlLoader.setController(this);

        try {
            fxmlLoader.load();
        } catch (IOException exception) {

        }
        parameters = new ArrayList<>();
        parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
        parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
        parameters.add(new PLCOperand(PLCDataType.INT, "In1"));
        parameters.add(new PLCOperand(PLCDataType.INT, "In2"));
        parameters.add(new PLCOperand(PLCDataType.INT, "Dest"));
        fillDragOver = new BackgroundFill(Color.BLUE, CornerRadii.EMPTY,
        Insets.EMPTY);
    }

    /**
     * Initializes the controller class.
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        buildHandlers();
    }

    public void relocateTo(Point2D p) {

```

```

        Point2D point = getParent().sceneToLocal(p);
        relocate(point.getX(), point.getY());
    }

    private void buildHandlers() {

        dragOver = new EventHandler<DragEvent>() {
            @Override
            public void handle(DragEvent event) {
                if (event.getGestureSource() != this
                    && event.getDragboard().hasString()) {
                    /* allow for moving */
                    event.acceptTransferModes(TransferMode.COPY);
                }
                event.consume();
            }
        };

        dragDropped = new EventHandler<DragEvent>() {
            @Override
            public void handle(DragEvent event) {
                if (event.getGestureTarget() != this) {
                    /* data dropped */
/* if there is a string data on dragboard, read it and use it */
                    Dragboard db = event.getDragboard();
                    boolean success = false;
                    if (db.hasString()) {

                        String data = db.getString();
                        String dataType = DragDataFormatter.getTagType(data);
                        if (validType(dataType)) {
                            String name = DragDataFormatter.getTagName(data);
                            if (event.getTarget().equals(in1TF)) {
                                in1TF.setText(name);
                            } else if (event.getTarget().equals(in2TF)) {
                                in2TF.setText(name);
                            } else if (event.getTarget().equals(destTF)) {
                                destTF.setText(name);
                            }
                        }
                    }
                    /* let the source know whether the string was
successfully
* transferred and used */
                    event.setDropCompleted(success);

                    event.consume();
                }
            }
        };

        dragEntered = new EventHandler<DragEvent>() {
            @Override
            public void handle(DragEvent event) {
                if (event.getGestureSource() != this
                    && event.getDragboard().hasString()) {
                    if (event.getTarget().equals(in1TF)) {

```

```

                in1TF.setBackground(new Background(fillDragOver));
            }
            if (event.getTarget().equals(in2TF)) {
                in2TF.setBackground(new Background(fillDragOver));
            }
            if (event.getTarget().equals(destTF)) {
                destTF.setBackground(new Background(fillDragOver));
            }
        }
        event.consume();
    }
};

dragExited = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            if (event.getTarget().equals(in1TF)) {
                in1TF.setBackground(Background.EMPTY);
            } else if (event.getTarget().equals(in2TF)) {
                in2TF.setBackground(Background.EMPTY);
            } else if (event.getTarget().equals(destTF)) {
                destTF.setBackground(Background.EMPTY);
            }
        }
        event.consume();
    }
};

in1TF.setOnDragOver(dragOver);
in1TF.setOnDragDropped(dragDropped);
in1TF.setOnDragEntered(dragEntered);
in1TF.setOnDragExited(dragExited);
in2TF.setOnDragOver(dragOver);
in2TF.setOnDragDropped(dragDropped);
in2TF.setOnDragEntered(dragEntered);
in2TF.setOnDragExited(dragExited);
destTF.setOnDragOver(dragOver);
destTF.setOnDragDropped(dragDropped);
destTF.setOnDragEntered(dragEntered);
destTF.setOnDragExited(dragExited);
}

@Override
public AnchorPane getInstance() {
    return this;
}

@Override
public PLCInstruction getPLCInstruction() {
    parameters.get(2).setIdentity(in1TF.getText());
    parameters.get(3).setIdentity(in2TF.getText());
    parameters.get(4).setIdentity(destTF.getText());
    PLCInstruction instruction = new PLCInstruction(PLCOperator.ADD);
    instruction.setOperands(parameters);
    return instruction;
}
}

```

```

        private boolean validType(String type) {
            return type.equals(DragDataFormatter.TagType_INT) ||
type.equals(DragDataFormatter.TagType_BYTE) ||
type.equals(DragDataFormatter.TagType_FLOAT);
        }

    }

package com.vts.jplceditor.ld.instruction.math;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOperand;
import com.vts.jplceditor.compiler.plc.PLCOperator;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.Instruction;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Insets;
import javafx.geometry.Point2D;
import javafx.scene.control.TextField;
import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.CornerRadii;
import javafx.scene.paint.Color;

/**
 * FXML Controller class
 *
 * @author Vusman
 */
public class Div extends AnchorPane implements Initializable, Instruction {

    @FXML
    private TextField in2TF;
    @FXML
    private TextField in1TF;
    @FXML
    private TextField destTF;
    private EventHandler<DragEvent> dragDropped;
    private EventHandler<DragEvent> dragOver;
    private EventHandler<DragEvent> dragEntered;
    private EventHandler<DragEvent> dragExited;
}

```

```

private Point2D dragOffset;
private boolean entered;
private final List<PLCOperand> parameters;
private final BackgroundFill fillDragOver;

public Div() {
    FXMLLoader fxmlLoader = new FXMLLoader(
        getClass().getResource(IR.Path_DIV));
    fxmlLoader.setRoot(this);
    fxmlLoader.setController(this);

    try {
        fxmlLoader.load();
    } catch (IOException exception) {

    }
    parameters = new ArrayList<>();
    parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
    parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
    parameters.add(new PLCOperand(PLCDataType.INT, "In1"));
    parameters.add(new PLCOperand(PLCDataType.INT, "In2"));
    parameters.add(new PLCOperand(PLCDataType.INT, "Dest"));
    fillDragOver = new BackgroundFill(Color.BLUE, CornerRadii.EMPTY,
    Insets.EMPTY);
}

/**
 * Initializes the controller class.
 */
@Override
public void initialize(URL url, ResourceBundle rb) {
    buildHandlers();
}

public void relocateTo(Point2D p) {
    Point2D point = getParent().sceneToLocal(p);
    relocate(point.getX(), point.getY());
}

private void buildHandlers() {

    dragOver = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                /* allow for moving */
                event.acceptTransferModes(TransferMode.COPY);
            }
            event.consume();
        }
    };
    dragDropped = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureTarget() != this) {

```

```

        /* data dropped */
/* if there is a string data on dragboard, read it and use it */
    Dragboard db = event.getDragboard();
    boolean success = false;
    if (db.hasString()) {

        String data = db.getString();
        String dataType = DragDataFormatter.getTagType(data);
        if (validType(dataType)) {
            String name = DragDataFormatter.getTagName(data);
            if (event.getTarget().equals(in1TF)) {
                in1TF.setText(name);
            } else if (event.getTarget().equals(in2TF)) {
                in2TF.setText(name);
            } else if (event.getTarget().equals(destTF)) {
                destTF.setText(name);
            }
        }
    }
    /* let the source know whether the string was
successfully
* transferred and used */
    event.setDropCompleted(success);

    event.consume();
}
};

dragEntered = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            if (event.getTarget().equals(in1TF)) {
                in1TF.setBackground(new Background(fillDragOver));
            }
            if (event.getTarget().equals(in2TF)) {
                in2TF.setBackground(new Background(fillDragOver));
            }
            if (event.getTarget().equals(destTF)) {
                destTF.setBackground(new Background(fillDragOver));
            }
        }
        event.consume();
    }
};
dragExited = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            if (event.getTarget().equals(in1TF)) {
                in1TF.setBackground(Background.EMPTY);
            } else if (event.getTarget().equals(in2TF)) {
                in2TF.setBackground(Background.EMPTY);
            } else if (event.getTarget().equals(destTF)) {

```

```

        destTF.setBackground(Background.EMPTY);
    }
}
event.consume();
}
};

in1TF.setOnDragOver(dragOver);
in1TF.setOnDragDropped(dragDropped);
in1TF.setOnDragEntered(dragEntered);
in1TF.setOnDragExited(dragExited);
in2TF.setOnDragOver(dragOver);
in2TF.setOnDragDropped(dragDropped);
in2TF.setOnDragEntered(dragEntered);
in2TF.setOnDragExited(dragExited);
destTF.setOnDragOver(dragOver);
destTF.setOnDragDropped(dragDropped);
destTF.setOnDragEntered(dragEntered);
destTF.setOnDragExited(dragExited);
}

@Override
public AnchorPane getInstance() {
    return this;
}

@Override
public PLCInstruction getPLCInstruction() {
    parameters.get(2).setIdentity(in1TF.getText());
    parameters.get(3).setIdentity(in2TF.getText());
    parameters.get(4).setIdentity(destTF.getText());
    PLCInstruction instruction = new PLCInstruction(PLCOperator.DIV);
    instruction.setOperands(parameters);
    return instruction;
}

private boolean validType(String type) {
    return type.equals(DragDataFormatter.TagType_INT) ||
    type.equals(DragDataFormatter.TagType_BYTE) ||
    type.equals(DragDataFormatter.TagType_FLOAT);
}

}

```

```

package com.vts.jplceditor.ld.instruction.math;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.compiler.plc.PLCOperator;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.Instruction;
import java.io.IOException;

```

```

import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Insets;
import javafx.geometry.Point2D;
import javafx.scene.control.TextField;
import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.CornerRadii;
import javafx.scene.paint.Color;

/**
 * FXML Controller class
 *
 * @author Vusman
 */
public class Mul extends AnchorPane implements Initializable, Instruction {

    @FXML
    private TextField in2TF;
    @FXML
    private TextField in1TF;
    @FXML
    private TextField destTF;
    private EventHandler<DragEvent> dragDropped;
    private EventHandler<DragEvent> dragOver;
    private EventHandler<DragEvent> dragEntered;
    private EventHandler<DragEvent> dragExited;
    private Point2D dragOffset;
    private boolean entered;
    private final List<PLCOperand> parameters;
    private final BackgroundFill fillDragOver;

    public Mul() {
        FXMLLoader fxmlLoader = new FXMLLoader(
            getClass().getResource(IR.Path_MUL));
        fxmlLoader.setRoot(this);
        fxmlLoader.setController(this);

        try {
            fxmlLoader.load();
        } catch (IOException exception) {

        }
        parameters = new ArrayList<>();
        parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
        parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
        parameters.add(new PLCOperand(PLCDataType.INT, "In1"));
    }
}

```

```

parameters.add(new PLCOperand(PLCDataType.INT, "In2"));
parameters.add(new PLCOperand(PLCDataType.INT, "Dest"));
fillDragOver = new BackgroundFill(Color.BLUE, CornerRadii.EMPTY,
Insets.EMPTY);
}

/**
 * Initializes the controller class.
 */
@Override
public void initialize(URL url, ResourceBundle rb) {
    buildHandlers();
}

public void relocateTo(Point2D p) {
    Point2D point = getParent().sceneToLocal(p);
    relocate(point.getX(), point.getY());
}

private void buildHandlers() {

    dragOver = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                /* allow for moving */
                event.acceptTransferModes(TransferMode.COPY);
            }
            event.consume();
        }
    };

    dragDropped = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureTarget() != this) {
                /* data dropped */
/* if there is a string data on dragboard, read it and use it */
                Dragboard db = event.getDragboard();
                boolean success = false;
                if (db.hasString()) {

                    String data = db.getString();
                    String dataType = DragDataFormatter.getTagType(data);
                    if (validType(dataType)) {
                        String name = DragDataFormatter.getTagName(data);
                        if (event.getTarget().equals(in1TF)) {
                            in1TF.setText(name);
                        } else if (event.getTarget().equals(in2TF)) {
                            in2TF.setText(name);
                        } else if (event.getTarget().equals(destTF)) {
                            destTF.setText(name);
                        }
                    }
                }
            }
        }
    };
}

```

```

        /* let the source know whether the string was
successfully
* transferred and used */
        event.setDropCompleted(success);

        event.consume();
    }
}

};

dragEntered = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            if (event.getTarget().equals(in1TF)) {
                in1TF.setBackground(new Background(fillDragOver));
            }
            if (event.getTarget().equals(in2TF)) {
                in2TF.setBackground(new Background(fillDragOver));
            }
            if (event.getTarget().equals(destTF)) {
                destTF.setBackground(new Background(fillDragOver));
            }
        }
        event.consume();
    }
};

dragExited = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            if (event.getTarget().equals(in1TF)) {
                in1TF.setBackground(Background.EMPTY);
            } else if (event.getTarget().equals(in2TF)) {
                in2TF.setBackground(Background.EMPTY);
            } else if (event.getTarget().equals(destTF)) {
                destTF.setBackground(Background.EMPTY);
            }
        }
        event.consume();
    }
};

in1TF.setOnDragOver(dragOver);
in1TF.setOnDragDropped(dragDropped);
in1TF.setOnDragEntered(dragEntered);
in1TF.setOnDragExited(dragExited);
in2TF.setOnDragOver(dragOver);
in2TF.setOnDragDropped(dragDropped);
in2TF.setOnDragEntered(dragEntered);
in2TF.setOnDragExited(dragExited);
destTF.setOnDragOver(dragOver);
destTF.setOnDragDropped(dragDropped);
destTF.setOnDragEntered(dragEntered);
destTF.setOnDragExited(dragExited);

```

```
}

@Override
public AnchorPane getInstance() {
    return this;
}

@Override
public PLCInstruction getPLCInstruction() {
    parameters.get(2).setIdentity(in1TF.getText());
    parameters.get(3).setIdentity(in2TF.getText());
    parameters.get(4).setIdentity(destTF.getText());
    PLCInstruction instruction = new PLCInstruction(PLCOperator.MUL);
    instruction.setOperands(parameters);
    return instruction;
}

private boolean validType(String type) {
    return type.equals(DragDataFormatter.TagType_INT) ||
    type.equals(DragDataFormatter.TagType_BYTE) ||
    type.equals(DragDataFormatter.TagType_FLOAT);
}

}
```

```
package com.vts.jplceditor.ld.instruction.math;

import com.vts.jplceditor.compiler.plc.PLCDATAType;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.compiler.plc.PLCOperator;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.Instruction;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Insets;
import javafx.geometry.Point2D;
import javafx.scene.control.TextField;
import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.CornerRadii;
import javafx.scene.paint.Color;
```

```

/**
 * FXML Controller class
 *
 * @author Vusman
 */
public class Sub extends AnchorPane implements Initializable, Instruction {

    @FXML
    private TextField in2TF;
    @FXML
    private TextField in1TF;
    @FXML
    private TextField destTF;
    private EventHandler<DragEvent> dragDropped;
    private EventHandler<DragEvent> dragOver;
    private EventHandler<DragEvent> dragEntered;
    private EventHandler<DragEvent> dragExited;
    private Point2D dragOffset;
    private boolean entered;
    private final List<PLCOperand> parameters;
    private final BackgroundFill fillDragOver;

    public Sub() {
        FXMLLoader fxmlLoader = new FXMLLoader(
            getClass().getResource(IR.Path_SUB));
        fxmlLoader.setRoot(this);
        fxmlLoader.setController(this);

        try {
            fxmlLoader.load();
        } catch (IOException exception) {

        }
        parameters = new ArrayList<>();
        parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
        parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
        parameters.add(new PLCOperand(PLCDataType.INT, "In1"));
        parameters.add(new PLCOperand(PLCDataType.INT, "In2"));
        parameters.add(new PLCOperand(PLCDataType.INT, "Dest"));
        fillDragOver = new BackgroundFill(Color.BLUE, CornerRadii.EMPTY,
        Insets.EMPTY);
    }

    /**
     * Initializes the controller class.
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        buildHandlers();
    }

    public void relocateTo(Point2D p) {
        Point2D point = getParent().sceneToLocal(p);
        relocate(point.getX(), point.getY());
    }
}

```

```

private void buildHandlers() {
    dragOver = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                /* allow for moving */
                event.acceptTransferModes(TransferMode.COPY);
            }
            event.consume();
        }
    };

    dragDropped = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureTarget() != this) {
                /* data dropped */
            }
            /* if there is a string data on dragboard, read it and use it */
            Dragboard db = event.getDragboard();
            boolean success = false;
            if (db.hasString()) {

                String data = db.getString();
                String dataType = DragDataFormatter.getTagType(data);
                if (validType(dataType)) {
                    String name = DragDataFormatter.getTagName(data);
                    if (event.getTarget().equals(in1TF)) {
                        in1TF.setText(name);
                    } else if (event.getTarget().equals(in2TF)) {
                        in2TF.setText(name);
                    } else if (event.getTarget().equals(destTF)) {
                        destTF.setText(name);
                    }
                }
            }
            /* let the source know whether the string was
            successfully
            * transferred and used */
            event.setDropCompleted(success);

            event.consume();
        }
    };

    dragEntered = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                if (event.getTarget().equals(in1TF)) {
                    in1TF.setBackground(new Background(fillDragOver));
                }
                if (event.getTarget().equals(in2TF)) {
                    in2TF.setBackground(new Background(fillDragOver));
                }
            }
        }
    };
}

```

```

        }
        if (event.getTarget().equals(destTF)) {
            destTF.setBackground(new Background(fillDragOver));
        }
    }
    event.consume();
}
};

dragExited = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            if (event.getTarget().equals(in1TF)) {
                in1TF.setBackground(Background.EMPTY);
            } else if (event.getTarget().equals(in2TF)) {
                in2TF.setBackground(Background.EMPTY);
            } else if (event.getTarget().equals(destTF)) {
                destTF.setBackground(Background.EMPTY);
            }
        }
        event.consume();
    }
};

in1TF.setOnDragOver(dragOver);
in1TF.setOnDragDropped(dragDropped);
in1TF.setOnDragEntered(dragEntered);
in1TF.setOnDragExited(dragExited);
in2TF.setOnDragOver(dragOver);
in2TF.setOnDragDropped(dragDropped);
in2TF.setOnDragEntered(dragEntered);
in2TF.setOnDragExited(dragExited);
destTF.setOnDragOver(dragOver);
destTF.setOnDragDropped(dragDropped);
destTF.setOnDragEntered(dragEntered);
destTF.setOnDragExited(dragExited);
}

@Override
public AnchorPane getInstance() {
    return this;
}

@Override
public PLCInstruction getPLCInstruction() {
    parameters.get(2).setIdentity(in1TF.getText());
    parameters.get(3).setIdentity(in2TF.getText());
    parameters.get(4).setIdentity(destTF.getText());
    PLCInstruction instruction = new PLCInstruction(PLCOperator.SUB);
    instruction.setOperands(parameters);
    return instruction;
}

private boolean validType(String type) {

```

```

        return type.equals(DragDataFormatter.TagType_INT) ||
type.equals(DragDataFormatter.TagType_BYTE) ||
type.equals(DragDataFormatter.TagType_FLOAT);
    }

}

package com.vts.jplceditor.ld.instruction.relay;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOperator;
import com.vts.jplceditor.compiler.plc.PLCOperand;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.control.Label;
import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Line;
import javafx.scene.shape.Rectangle;
import com.vts.jplceditor.ld.instruction.Instruction;

/**
 *
 * @author Vusumuzi_Tshabangu
 */
public class Ote extends AnchorPane implements Instruction, Initializable {

    @FXML
    Label tagName;
    @FXML
    private Rectangle dragOverIndicator;
    @FXML
    private Line left_line;
    @FXML
    private Line right_line;
    @FXML
    private Line left_bar;
    @FXML
    private Line right_bar;
    @FXML
    private Line center_bar;
}

```

```

private EventHandler<DragEvent> dragDropped;
private EventHandler<DragEvent> dragOver;
private EventHandler<DragEvent> dragEntered;
private EventHandler<DragEvent> dragExited;
private Point2D dragOffset;
private boolean entered;
private List<PLCOperand> parameters;

/**
 *
 */
public Ote() {
    FXMLLoader fxmlLoader = new FXMLLoader(
        getClass().getResource(IR.Path_OTE));
    fxmlLoader.setRoot(this);
    fxmlLoader.setController(this);

    try {
        fxmlLoader.load();
    } catch (IOException exception) {

    }
    parameters = new ArrayList<>();
    parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
    parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
    parameters.add(new PLCOperand(PLCDataType.BIT, tagName.getText()));
}
}

@Override
public void initialize(URL location, ResourceBundle resources) {
    buildHandlers();
}

public void relocateTo(Point2D p) {
    Point2D point = getParent().sceneToLocal(p);
    relocate(point.getX(), point.getY());
}

private void buildHandlers() {

    dragOver = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                /* allow for moving */
                event.acceptTransferModes(TransferMode.COPY);
                if (entered) {
                    relocateTo(new Point2D(event.getSceneX() -
dragOffset.getX(),
                                event.getSceneY() - dragOffset.getY()));
                }
            }
            event.consume();
        }
    };
}

```

```

dragDropped = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureTarget() == this) {
            entered = false;
        }
        /* data dropped */
    /* if there is a string data on dragboard, read it and use it */
    Dragboard db = event.getDragboard();
    boolean success = false;
    if (db.hasString()) {
        String data = db.getString();
        String dataType = DragDataFormatter.getTagType(data);
        if (dataType.equals(DragDataFormatter.TagType_BIT)) {
            String name = DragDataFormatter.getTagName(data);
            tagName.setText(name);
        }
    }
    /* let the source know whether the string was successfully
     * transferred and used */
    event.setDropCompleted(success);

    event.consume();
}
};

dragEntered = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            dragOverIndicator.setFill(Color.BLUE);
        }
        event.consume();
    }
};
dragExited = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            dragOverIndicator.setFill(Color.WHITE);
        }
        event.consume();
    }
};

setOnDragOver(dragOver);
setOnDragDropped(dragDropped);
setOnDragEntered(dragEntered);
setOnDragExited(dragExited);
}

@Override
public AnchorPane getInstance() {
    return this;
}
}

```

```

@Override
public PLCInstruction getPLCInstruction() {
    parameters.get(2).setIdentity(tagName.getText());
    PLCInstruction instruction = new PLCInstruction(PLCOperator.OTE);
    instruction.setOperands(parameters);
    return instruction;
}
}

package com.vts.jplceditor.ld.instruction.relay;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOperator;
import com.vts.jplceditor.compiler.plc.PLCOperand;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Point2D;
import javafx.scene.Node;
import javafx.scene.control.Label;
import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Line;
import javafx.scene.shape.Rectangle;
import com.vts.jplceditor.ld.instruction.Instruction;

/**
 *
 * @author Vusumuzi_Tshabangu
 */
public class Otl extends AnchorPane implements Instruction, Initializable {

    @FXML
    Label tagName;
    @FXML
    private Rectangle dragOverIndicator;
    @FXML
    private Line left_line;
    @FXML
    private Line right_line;
    @FXML
}

```

```

private Line left_bar;
@FXML
private Line right_bar;
@FXML
private Line center_bar;

private EventHandler<DragEvent> dragDropped;
private EventHandler<DragEvent> dragOver;
private EventHandler<DragEvent> dragEntered;
private EventHandler<DragEvent> dragExited;
private Point2D dragOffset;
private boolean entered;
private List<PLCOperand> parameters;

/**
 *
 */
public Otl() {
    FXMLLoader fxmlLoader = new FXMLLoader(
        getClass().getResource(IR.Path_OTL));
    fxmlLoader.setRoot(this);
    fxmlLoader.setController(this);

    try {
        fxmlLoader.load();
    } catch (IOException exception) {

    }
    parameters = new ArrayList<>();
    parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
    parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
    parameters.add(new PLCOperand(PLCDataType.BIT, tagName.getText()));
}
@Override
public void initialize(URL location, ResourceBundle resources) {
    buildHandlers();
}

public void relocateTo(Point2D p) {
    Point2D point = getParent().sceneToLocal(p);
    relocate(point.getX(), point.getY());
}

private void buildHandlers() {

    dragOver = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                /* allow for moving */
                event.acceptTransferModes(TransferMode.COPY);
                if (entered) {
                    relocateTo(new Point2D(event.getSceneX() -
dragOffset.getX(),
event.getSceneY() - dragOffset.getY())));
                }
            }
        }
    };
}

```

```

        }
    }
    event.consume();
}
};

dragDropped = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureTarget() == this) {
            entered = false;
        }
        /* data dropped */
/* if there is a string data on dragboard, read it and use it */
        Dragboard db = event.getDragboard();
        boolean success = false;
        if (db.hasString()) {
            String data = db.getString();
            String dataType = DragDataFormatter.getTagType(data);
            if (dataType.equals(DragDataFormatter.TagType_BIT)) {
                String name = DragDataFormatter.getTagName(data);
                tagName.setText(name);
            }
        }
        /* let the source know whether the string was successfully
 * transferred and used */
        event.setDropCompleted(success);
        event.consume();
    }
};

dragEntered = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            dragOverIndicator.setFill(Color.BLUE);
        }
        event.consume();
    }
};

dragExited = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            dragOverIndicator.setFill(Color.WHITE);
        }
        event.consume();
    }
};

setOnDragOver(dragOver);
setOnDragDropped(dragDropped);
setOnDragEntered(dragEntered);
setOnDragExited(dragExited);

```

```

    }

    @Override
    public AnchorPane getInstance() {
        return this;
    }

    @Override
    public PLCInstruction getPLCInstruction() {
        parameters.get(2).setIdentity(tagName.getText());
        PLCInstruction instruction = new PLCInstruction(PLCOperator.OTL);
        instruction.setOperands(parameters);
        return instruction;
    }
}

package com.vts.jplceditor.ld.instruction.relay;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOperator;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.control.Label;
import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Line;
import javafx.scene.shape.Rectangle;
import com.vts.jplceditor.ld.instruction.Instruction;

/**
 * @author Vusumuzi_Tshabangu
 */
public class Otu extends AnchorPane implements Instruction, Initializable {
    @FXML
    Label tagName;
    @FXML
}

```

```

private Rectangle dragOverIndicator;
@FXML
private Line left_line;
@FXML
private Line right_line;
@FXML
private Line left_bar;
@FXML
private Line right_bar;
@FXML
private Line center_bar;

private EventHandler<DragEvent> dragDropped;
private EventHandler<DragEvent> dragOver;
private EventHandler<DragEvent> dragEntered;
private EventHandler<DragEvent> dragExited;
private Point2D dragOffset;
private boolean entered;
private List<PLCOperand> parameters;

/**
 *
 */
public Otu() {
    FXMLLoader fxmlLoader = new FXMLLoader(
        getClass().getResource(IR.Path_OTU));
    fxmlLoader.setRoot(this);
    fxmlLoader.setController(this);

    try {
        fxmlLoader.load();
    } catch (IOException exception) {

    }
    parameters = new ArrayList<>();
    parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
    parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
    parameters.add(new PLCOperand(PLCDataType.BIT, tagName.getText()));
}

@Override
public void initialize(URL location, ResourceBundle resources) {
    buildHandlers();
}

public void relocateTo(Point2D p) {
    Point2D point = getParent().sceneToLocal(p);
    relocate(point.getX(), point.getY());
}

private void buildHandlers() {

    dragOver = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
}

```

```

        /* allow for moving */
        event.acceptTransferModes(TransferMode.COPY);
        if (entered) {
            relocateTo(new Point2D(event.getSceneX() -
dragOffset.getX(), event.getSceneY() - dragOffset.getY())));
        }
    }
    event.consume();
}
};

dragDropped = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureTarget() == this) {
            entered = false;
        }
        /* data dropped */
        /* if there is a string data on dragboard, read it and use it */
        Dragboard db = event.getDragboard();
        boolean success = false;
        if (db.hasString()) {
            String data = db.getString();
            String dataType = DragDataFormatter.getTagType(data);
            if (dataType.equals(DragDataFormatter.TagType_BIT)) {
                String name = DragDataFormatter.getTagName(data);
                tagName.setText(name);
            }
        }
        /* let the source know whether the string was successfully
 * transferred and used */
        event.setDropCompleted(success);

        event.consume();
    }
};

dragEntered = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            dragOverIndicator.setFill(Color.BLUE);
        }
    }
    event.consume();
}
};

dragExited = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            dragOverIndicator.setFill(Color.WHITE);
        }
    }
    event.consume();
}
};

```

```
};

setOnDragOver(dragOver);
setOnDragDropped(dragDropped);
setOnDragEntered(dragEntered);
setOnDragExited(dragExited);
}

@Override
public AnchorPane getInstance() {
    return this;
}

@Override
public PLCInstruction getPLCInstruction() {
    parameters.get(2).setIdentity(tagName.getText());
    PLCInstruction instruction = new PLCInstruction(PLCOperator.OTU);
    instruction.setOperands(parameters);
    return instruction;
}
}
```

```
package com.vts.jplceditor.ld.instruction.relay;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOoperator;
import com.vts.jplceditor.compiler.plc.PLCOoperand;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Point2D;
import javafx.scene.control.Label;
import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Line;
import javafx.scene.shape.Rectangle;
import com.vts.jplceditor.ld.instruction.Instruction;

/**
 *
 * @author Vusumuzi_Tshabangu
 */

```

```
public class Xic extends AnchorPane implements Instruction, Initializable {

    @FXML
    Label tagName;
    @FXML
    private Rectangle dragOverIndicator;
    @FXML
    private Line left_line;
    @FXML
    private Line right_line;
    @FXML
    private Line left_bar;
    @FXML
    private Line right_bar;
    @FXML
    private Line center_bar;

    private EventHandler<DragEvent> dragDropped;
    private EventHandler<DragEvent> dragOver;
    private EventHandler<DragEvent> dragEntered;
    private EventHandler<DragEvent> dragExited;
    private Point2D dragOffset;
    private boolean entered;
    private final List<PLCOperand> parameters;

    /**
     */
    public Xic() {
        FXMLLoader fxmlLoader = new FXMLLoader(
            getClass().getResource(IR.Path_XIC));
        fxmlLoader.setRoot(this);
        fxmlLoader.setController(this);

        try {
            fxmlLoader.load();
        } catch (IOException exception) {

        }
        parameters = new ArrayList<>();
        parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
        parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
        parameters.add(new PLCOperand(PLCDataType.BIT, tagName.getText()));
    }

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        buildHandlers();
    }

    public void relocateTo(Point2D p) {
        Point2D point = getParent().sceneToLocal(p);
        relocate(point.getX(), point.getY());
    }

    private void buildHandlers() {
```

```

dragOver = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            /* allow for moving */
            event.acceptTransferModes(TransferMode.COPY);
        if (entered) {
            relocateTo(new Point2D(event.getSceneX() -
dragOffset.getX(),
                           event.getSceneY() - dragOffset.getY()));
        }
    }
    event.consume();
}
};

dragDropped = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureTarget() == this) {
            entered = false;
        }
        /* data dropped */
/* if there is a string data on dragboard, read it and use it */
        Dragboard db = event.getDragboard();
        boolean success = false;
        if (db.hasString()) {
            String data = db.getString();
            String dataType = DragDataFormatter.getTagType(data);
            if (dataType.equals(DragDataFormatter.TagType_BIT)) {
                String name = DragDataFormatter.getTagName(data);
                tagName.setText(name);
            }
        }
        /* let the source know whether the string was successfully
* transferred and used */
        event.setDropCompleted(success);

        event.consume();
}
};

dragEntered = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this
            && event.getDragboard().hasString()) {
            dragOverIndicator.setFill(Color.BLUE);
        }
        event.consume();
}
};

dragExited = new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        if (event.getGestureSource() != this

```

```
        && event.getDragboard().hasString()) {
            dragOverIndicator.setFill(Color.WHITE);
        }
        event.consume();
    }
};

setOnDragOver(dragOver);
setOnDragDropped(dragDropped);
setOnDragEntered(dragEntered);
setOnDragExited(dragExited);
}

@Override
public AnchorPane getInstance() {
    return this;
}

@Override
public PLCInstruction getPLCInstruction() {
    parameters.get(2).setIdentity(tagName.getText());
    PLCInstruction instruction = new PLCInstruction(PLCOperator.XIC);
    instruction.setOperands(parameters);
    return instruction;
}
}
```

```
package com.vts.jplceditor.ld.instruction.relay;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOperator;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.ld.instruction.IR;
import com.vts.jplceditor.ld.instruction.DragDataFormatter;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Point2D;
import javafx.scene.Node;
import javafx.scene.control.Label;
import javafx.scene.input.DragEvent;
import javafx.scene.input.Dragboard;
import javafx.scene.input.TransferMode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Line;
import javafx.scene.shape.Rectangle;
```

```
import com.vts.jplceditor.ld.instruction.Instruction;

/**
 *
 * @author Vusumuzi_Tshabangu
 */
public class Xio extends AnchorPane implements Instruction, Initializable {

    @FXML
    Label tagName;
    @FXML
    private Rectangle dragOverIndicator;
    @FXML
    private Line left_line;
    @FXML
    private Line right_line;
    @FXML
    private Line left_bar;
    @FXML
    private Line right_bar;
    @FXML
    private Line center_bar;

    private EventHandler<DragEvent> dragDropped;
    private EventHandler<DragEvent> dragOver;
    private EventHandler<DragEvent> dragEntered;
    private EventHandler<DragEvent> dragExited;
    private Point2D dragOffset;
    private boolean entered;
    private List<PLCOperand> parameters;

    /**
     *
     */
    public Xio() {
        FXMLLoader fxmlLoader = new FXMLLoader(
            getClass().getResource(IR.Path_XIO));
        fxmlLoader.setRoot(this);
        fxmlLoader.setController(this);

        try {
            fxmlLoader.load();
        } catch (IOException exception) {

        }
        parameters = new ArrayList<>();
        parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
        parameters.add(new PLCOperand(PLCDataType.BIT, "ENO"));
        parameters.add(new PLCOperand(PLCDataType.BIT, tagName.getText()));
    }

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        buildHandlers();
    }

    public void relocateTo(Point2D p) {
```

```

        Point2D point = getParent().sceneToLocal(p);
        relocate(point.getX(), point.getY());
    }

    private void buildHandlers() {
        dragOver = new EventHandler<DragEvent>() {
            @Override
            public void handle(DragEvent event) {
                if (event.getGestureSource() != this
                    && event.getDragboard().hasString()) {
                    /* allow for moving */
                    event.acceptTransferModes(TransferMode.COPY);
                    if (entered) {
                        relocateTo(new Point2D(event.getSceneX() -
dragOffset.getX(),
                                event.getSceneY() - dragOffset.getY())));
                    }
                }
                event.consume();
            }
        };
        dragDropped = new EventHandler<DragEvent>() {
            @Override
            public void handle(DragEvent event) {
                if (event.getGestureTarget() == this) {
                    entered = false;
                }
                /* data dropped */
                /* if there is a string data on dragboard, read it and use it */
                Dragboard db = event.getDragboard();
                boolean success = false;
                if (db.hasString()) {
                    String data = db.getString();
                    String dataType = DragDataFormatter.getTagType(data);
                    if (dataType.equals(DragDataFormatter.TagType_BIT)) {
                        String name = DragDataFormatter.getTagName(data);
                        tagName.setText(name);
                    }
                }
                /* let the source know whether the string was successfully
 * transferred and used */
                event.setDropCompleted(success);

                event.consume();
            }
        };
        dragEntered = new EventHandler<DragEvent>() {
            @Override
            public void handle(DragEvent event) {
                if (event.getGestureSource() != this
                    && event.getDragboard().hasString()) {
                    dragOverIndicator.setFill(Color.BLUE);
                }
                event.consume();
            }
        };
    }
}

```

```
        }

    dragExited = new EventHandler<DragEvent>() {
        @Override
        public void handle(DragEvent event) {
            if (event.getGestureSource() != this
                && event.getDragboard().hasString()) {
                dragOverIndicator.setFill(Color.WHITE);
            }
            event.consume();
        }
    };

    setOnDragOver(dragOver);
    setOnDragDropped(dragDropped);
    setOnDragEntered(dragEntered);
    setOnDragExited(dragExited);
}

@Override
public AnchorPane getInstance() {
    return this;
}

@Override
public PLCInstruction getPLCInstruction() {
    parameters.get(2).setIdentity(tagName.getText());
    PLCInstruction instruction = new PLCInstruction(PLCOperator.XIO);
    instruction.setOperands(parameters);
    return instruction;
}
```

```
package com.vts.jplceditor.ld.instruction.timer;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.compiler.plc.PLCOperator;
import com.vts.jplceditor.ld.instruction.IR;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Point2D;
import javafx.scene.control.ComboBox;
```

```

import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import com.vts.jplceditor.ld.instruction.Instruction;

/**
 * FXML Controller class
 *
 * @author Vusivus
 */
public class Ctd extends AnchorPane implements Initializable, Instruction {

    @FXML
    private ComboBox<String> timerCB;
    @FXML
    private TextField countTF;
    private boolean entered;
    private List<PLCOperand> parameters;
    private EventHandler<ActionEvent> cbAction;
    private EventHandler<ActionEvent> tfAction;

    public Ctd() {
        FXMLLoader fxmlLoader = new FXMLLoader(
            getClass().getResource(IR.Path_CTD));
        fxmlLoader.setRoot(this);
        fxmlLoader.setController(this);
        try {
            fxmlLoader.load();
        } catch (IOException exception) {

        }
    }

    /**
     * Initializes the controller class.
     *
     * @param url
     * @param rb
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        ObservableList<String> options
            = FXCollections.observableArrayList("Counter1",
                "Counter2");
        timerCB.setItems(options);
        parameters = new ArrayList<>();
        parameters.add(new PLCOperand(PLCDataType.BIT, "CD"));
        parameters.add(new PLCOperand(PLCDataType.BIT, "UN"));
        parameters.add(new PLCOperand(PLCDataType.BIT, "DN"));
        parameters.add(new PLCOperand(PLCDataType.BIT, "R"));
        parameters.add(new PLCOperand(PLCDataType.BYTE, "Counter1"));
        parameters.add(new PLCOperand(PLCDataType.INT, "PRE"));
        parameters.add(new PLCOperand(PLCDataType.INT, "ACC"));
        buildHandlers();
    }

    @Override

```

```

public AnchorPane getInstance() {
    return this;
}

public void relocateTo(Point2D p) {
    Point2D point = getParent().sceneToLocal(p);
    relocate(point.getX(), point.getY());
}

@Override
public PLCInstruction getPLCInstruction() {

parameters.get(4).setIdentity(timerCB.getSelectionModel().getSelectedItem());
parameters.get(5).setIdentity(countTF.getText());
PLCInstruction instruction = new PLCInstruction(PLCOperator.CTU);
instruction.setOperands(parameters);
return instruction;
}

private void buildHandlers() {
    cbAction = new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            if (timerCB.getSelectionModel().getSelectedIndex() >= 0) {
                countTF.setDisable(false);
                parameters.get(4).setIdentity((String)
timerCB.getSelectionModel().getSelectedItem());
            }
        }
    };
    tfAction = new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            String text = countTF.getText();
            int value = 0;
            try {
                value = Integer.parseInt(text);
            } catch (NumberFormatException ex) {

            };
            countTF.setText(String.valueOf(value));
            parameters.get(5).setIdentity(countTF.getText());
        }
    };
    timerCB.setOnAction(cbAction);
    countTF.setOnAction(tfAction);
}
}

package com.vts.jplceditor.ld.instruction.timer;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOperand;

```

```
import com.vts.jplceditor.compiler.plc.PLCOperator;
import com.vts.jplceditor.ld.instruction.IR;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Point2D;
import javafx.scene.control.ComboBox;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import com.vts.jplceditor.ld.instruction.Instruction;

/**
 * FXML Controller class
 *
 * @author Vusivus
 */
public class Ctu extends AnchorPane implements Initializable, Instruction {

    @FXML
    private ComboBox<String> timerCB;
    @FXML
    private TextField countTF;
    private boolean entered;
    private List<PLCOOperand> parameters;
    private EventHandler<ActionEvent> cbAction;
    private EventHandler<ActionEvent> tfAction;

    public Ctu() {
        FXMLLoader fxmlLoader = new FXMLLoader(
            getClass().getResource(IR.Path_CTU));
        fxmlLoader.setRoot(this);
        fxmlLoader.setController(this);
        try {
            fxmlLoader.load();
        } catch (IOException exception) {
        }
    }

    /**
     * Initializes the controller class.
     *
     * @param url
     * @param rb
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        ObservableList<String> options
```

```

        = FXCollections.observableArrayList("Counter1",
                "Counter2"
        );
    timerCB.setItems(options);
    parameters = new ArrayList<>();
    parameters.add(new PLCOperand(PLCDataType.BIT, "CU"));
    parameters.add(new PLCOperand(PLCDataType.BIT, "OV"));
    parameters.add(new PLCOperand(PLCDataType.BIT, "DN"));
    parameters.add(new PLCOperand(PLCDataType.BIT, "R"));
    parameters.add(new PLCOperand(PLCDataType.BYTE, "Counter1"));
    parameters.add(new PLCOperand(PLCDataType.INT, "PRE"));
    parameters.add(new PLCOperand(PLCDataType.INT, "ACC"));
    buildHandlers();
}

@Override
public AnchorPane getInstance() {
    return this;
}

public void relocateTo(Point2D p) {
    Point2D point = getParent().sceneToLocal(p);
    relocate(point.getX(), point.getY());
}

@Override
public PLCInstruction getPLCInstruction() {

parameters.get(4).setIdentity(timerCB.getSelectionModel().getSelectedItem());
parameters.get(5).setIdentity(countTF.getText());
PLCInstruction instruction = new PLCInstruction(PLCOperator.CTU);
instruction.setOperands(parameters);
return instruction;
}

private void buildHandlers() {
    cbAction = new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            if (timerCB.getSelectionModel().getSelectedIndex() >= 0) {
                countTF.setDisable(false);
                parameters.get(4).setIdentity((String)
timerCB.getSelectionModel().getSelectedItem());
            }
        }
    };
    tfAction = new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            String text = countTF.getText();
            int value = 0;
            try {
                value = Integer.parseInt(text);
            } catch (NumberFormatException ex) {

            };
            countTF.setText(String.valueOf(value));
        }
    };
}

```

```

        parameters.get(5).setIdentity(countTF.getText());
    }
}

timerCB.setOnAction(cbAction);
countTF.setOnAction(tfAction);
}

}

package com.vts.jplceditor.ld.instruction.timer;

import com.vts.jplceditor.compiler.plc.PLCDatatype;
import com.vts.jplceditor.compiler.plc.PLCInstruction;
import com.vts.jplceditor.compiler.plc.PLCOOperand;
import com.vts.jplceditor.compiler.plc.PLCOOperator;
import com.vts.jplceditor.ld.instruction.IR;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Point2D;
import javafx.scene.control.ComboBox;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import com.vts.jplceditor.ld.instruction.Instruction;

/**
 * FXML Controller class
 *
 * @author Vusivus
 */
public class Ton extends AnchorPane implements Initializable, Instruction {

    @FXML
    private ComboBox<String> timerCB;
    @FXML
    private TextField countTF;
    private boolean entered;
    private List<PLCOOperand> parameters;
    private EventHandler<ActionEvent> cbAction;
    private EventHandler<ActionEvent> tfAction;

    public Ton() {
        FXMLLoader fxmlLoader = new FXMLLoader(
            getClass().getResource(IR.Path.TON));

```

```

        fxmlLoader.setRoot(this);
        fxmlLoader.setController(this);
        try {
            fxmlLoader.load();
        } catch (IOException exception) {

        }
    }

    /**
     * Initializes the controller class.
     *
     * @param url
     * @param rb
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        ObservableList<String> options
            = FXCollections.observableArrayList("Timer1",
                "Timer2"
            );
        timerCB.setItems(options);
        parameters = new ArrayList<>();
        parameters.add(new PLCOperand(PLCDataType.BIT, "EN"));
        parameters.add(new PLCOperand(PLCDataType.BIT, "TT"));
        parameters.add(new PLCOperand(PLCDataType.BIT, "DN"));
        parameters.add(new PLCOperand(PLCDataType.BIT, "R"));
        parameters.add(new PLCOperand(PLCDataType.BYTE, "Timer1"));
        parameters.add(new PLCOperand(PLCDataType.INT, "PRE"));
        parameters.add(new PLCOperand(PLCDataType.INT, "ACC"));
        buildHandlers();
    }

    @Override
    public AnchorPane getInstance() {
        return this;
    }

    public void relocateTo(Point2D p) {
        Point2D point = getParent().sceneToLocal(p);
        relocate(point.getX(), point.getY());
    }

    @Override
    public PLCInstruction getPLCInstruction() {

        parameters.get(4).setIdentity(timerCB.getSelectionModel().getSelectedItem());
        parameters.get(5).setIdentity(countTF.getText());
        PLCInstruction instruction = new PLCInstruction(PLCOperator.TON);
        instruction.setOperands(parameters);
        return instruction;
    }

    private void buildHandlers() {
        cbAction = new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {

```

```
        if (timerCB.getSelectionModel().getSelectedIndex() >= 0) {
            countTF.setDisable(false);
            parameters.get(4).setIdentity((String)
timerCB.getSelectionModel().getSelectedItem());
        }
    };
tfAction = new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        String text = countTF.getText();
        int value = 0;
        try {
            value = Integer.parseInt(text);
        } catch (NumberFormatException ex) {
            ;
        }
        countTF.setText(String.valueOf(value));
        parameters.get(5).setIdentity(countTF.getText());
    }
};

timerCB.setOnAction(cbAction);
countTF.setOnAction(tfAction);
}
}
```

Ladder Diagram Instruction FXML codes

Equ.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="90.0" prefWidth="120.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1" >
    <children>
        <Label layoutX="10.0" layoutY="35.0" text="IN1">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="10.0" layoutY="65.0" text="IN2">
            <font>
                <Font size="9.0" />
```

```

        </font></Label>
    <Label layoutX="33.0" text="EQU" />
    <Line endY="90.0" startY="8.0" strokeWidth="3.0" />
    <Line endX="120.0" endY="90.0" startX="120.0" startY="8.0" 
strokeWidth="3.0" />
    <Line endX="120.0" endY="90.0" startY="90.0" strokeWidth="3.0" />
    <Line endX="30.0" endY="8.0" startY="8.0" strokeWidth="3.0" />
    <Line endX="120.0" endY="8.0" startX="62.0" startY="8.0" 
strokeWidth="3.0" />
<TextField fx:id="in2TF" disable="false" layoutX="40.0" layoutY="58.0" 
prefWidth="75.0" />
    <TextField fx:id="in1TF" disable="false" layoutX="40.0" layoutY="28.0" 
prefWidth="75.0" />
</children>
</fx:root>

```

Grt.fxml

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="90.0" prefWidth="120.0" 
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8" 
xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Label layoutX="10.0" layoutY="35.0" text="IN1">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="10.0" layoutY="65.0" text="IN2">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="33.0" text="GRT" />
        <Line endY="90.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="90.0" startX="120.0" startY="8.0" 
strokeWidth="3.0" />
        <Line endX="120.0" endY="90.0" startY="90.0" strokeWidth="3.0" />
        <Line endX="30.0" endY="8.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="8.0" startX="62.0" startY="8.0" 
strokeWidth="3.0" />
<TextField fx:id="in2TF" disable="false" layoutX="40.0" layoutY="58.0" 
prefWidth="75.0" />
    <TextField fx:id="in1TF" disable="false" layoutX="40.0" layoutY="28.0" 
prefWidth="75.0" />
</children>
</fx:root>

```

Les.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="90.0" prefWidth="120.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Label layoutX="10.0" layoutY="35.0" text="IN1">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="10.0" layoutY="65.0" text="IN2">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="33.0" text="LES" />
        <Line endY="90.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="90.0" startX="120.0" startY="8.0"
strokeWidth="3.0" />
        <Line endX="120.0" endY="90.0" startY="90.0" strokeWidth="3.0" />
        <Line endX="30.0" endY="8.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="8.0" startX="62.0" startY="8.0"
strokeWidth="3.0" />
        <TextField fx:id="in2TF" disable="false" layoutX="40.0" layoutY="58.0"
prefWidth="75.0" />
        <TextField fx:id="in1TF" disable="false" layoutX="40.0" layoutY="28.0"
prefWidth="75.0" />
    </children>
</fx:root>
```

Lim.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="130.0" prefWidth="120.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Label layoutX="10.0" layoutY="35.0" text="LOW">
            <font>
```

```

        <Font size="9.0" />
    </font></Label>
<Label layoutX="10.0" layoutY="68.0" text="TEST">
    <font>
        <Font size="9.0" />
    </font></Label>
<Label layoutX="33.0" text="LIM" />
<Line endY="130.0" startY="8.0" strokeWidth="3.0" />
<Line endX="120.0" endY="130.0" startX="120.0" startY="8.0"
strokeWidth="3.0" />
<Line endX="120.0" endY="130.0" startY="130.0" strokeWidth="3.0" />
<Line endX="30.0" endY="8.0" startY="8.0" strokeWidth="3.0" />
<Line endX="120.0" endY="8.0" startX="62.0" startY="8.0"
strokeWidth="3.0" />
<TextField fx:id="testTF" disable="false" layoutX="40.0" layoutY="63.0"
prefWidth="75.0" />
<TextField fx:id="lowTF" disable="false" layoutX="40.0" layoutY="28.0"
prefWidth="75.0" />
<TextField fx:id="highTF" disable="false" layoutX="40.0" layoutY="97.0"
prefWidth="75.0" />
    <Label layoutX="10.0" layoutY="102.0" text="HIGH">
        <font>
            <Font size="9.0" />
        </font>
    </Label>
</children>
</fx:root>
```

Mov.fxml

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="90.0" prefWidth="120.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Label layoutX="10.0" layoutY="35.0" text="Source">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="31.0" text="MOV" />
        <Line endY="90.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="90.0" startX="120.0" startY="8.0"
strokeWidth="3.0" />
        <Line endX="120.0" endY="90.0" startY="90.0" strokeWidth="3.0" />
        <Line endX="30.0" endY="8.0" startY="8.0" strokeWidth="3.0" />
```

```
<Line endX="120.0" endY="8.0" startX="62.0" startY="8.0"
strokeWidth="3.0" />
<TextField fx:id="sourceTF" disable="false" layoutX="40.0"
layoutY="28.0" prefWidth="75.0" />
<TextField fx:id="destTF" disable="false" layoutX="40.0" layoutY="60.0"
prefWidth="75.0" />
<Label layoutX="10.0" layoutY="65.0" text="Dest.">
<font>
<Font size="9.0" />
</font>
</Label>
</children>
</fx:root>
```

Mvm.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="120.0" prefWidth="120.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1">
<children>
<Label layoutX="10.0" layoutY="35.0" text="Source">
<font>
<Font size="9.0" />
</font></Label>
<Label layoutX="10.0" layoutY="65.0" text="Mask">
<font>
<Font size="9.0" />
</font></Label>
<Label layoutX="31.0" text="MVM" />
<Line endY="120.0" startY="8.0" strokeWidth="3.0" />
<Line endX="120.0" endY="120.0" startX="120.0" startY="8.0"
strokeWidth="3.0" />
<Line endX="120.0" endY="120.0" startY="120.0" strokeWidth="3.0" />
<Line endX="30.0" endY="8.0" startY="8.0" strokeWidth="3.0" />
<Line endX="120.0" endY="8.0" startX="62.0" startY="8.0"
strokeWidth="3.0" />
<TextField fx:id="maskTF" disable="false" layoutX="40.0" layoutY="58.0"
prefWidth="75.0" />
<TextField fx:id="sourceTF" disable="false" layoutX="40.0"
layoutY="28.0" prefWidth="75.0" />
<TextField fx:id="destTF" disable="false" layoutX="40.0" layoutY="87.0"
prefWidth="75.0" />
<Label layoutX="10.0" layoutY="93.0" text="Dest.">
<font>
<Font size="9.0" />
</font>
```

```
</Label>
</children>
</fx:root>
```

And.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="120.0" prefWidth="120.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Label layoutX="10.0" layoutY="35.0" text="IN1">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="10.0" layoutY="65.0" text="IN2">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="33.0" text="AND" />
        <Line endY="120.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="120.0" startX="120.0" startY="8.0"
strokeWidth="3.0" />
        <Line endX="120.0" endY="120.0" startY="120.0" strokeWidth="3.0" />
        <Line endX="30.0" endY="8.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="8.0" startX="62.0" startY="8.0"
strokeWidth="3.0" />
        <TextField fx:id="in2TF" disable="false" layoutX="40.0" layoutY="58.0"
prefWidth="75.0" />
        <TextField fx:id="in1TF" disable="false" layoutX="40.0" layoutY="28.0"
prefWidth="75.0" />
        <TextField fx:id="destTF" disable="false" layoutX="40.0" layoutY="87.0"
prefWidth="75.0" />
        <Label layoutX="10.0" layoutY="93.0" text="DEST">
            <font>
                <Font size="9.0" />
            </font>
        </Label>
    </children>
</fx:root>
```

Or.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="120.0" prefWidth="120.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Label layoutX="10.0" layoutY="35.0" text="IN1">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="10.0" layoutY="65.0" text="IN2">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="35.0" text="OR" />
        <Line endY="120.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="120.0" startX="120.0" startY="8.0"
strokeWidth="3.0" />
        <Line endX="120.0" endY="120.0" startY="120.0" strokeWidth="3.0" />
        <Line endX="30.0" endY="8.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="8.0" startX="62.0" startY="8.0"
strokeWidth="3.0" />
        <TextField fx:id="in2TF" disable="false" layoutX="40.0" layoutY="58.0"
prefWidth="75.0" />
        <TextField fx:id="in1TF" disable="false" layoutX="40.0" layoutY="28.0"
prefWidth="75.0" />
        <TextField fx:id="destTF" disable="false" layoutX="40.0" layoutY="87.0"
prefWidth="75.0" />
        <Label layoutX="10.0" layoutY="93.0" text="DEST">
            <font>
                <Font size="9.0" />
            </font>
        </Label>
    </children>
</fx:root>
```

Or.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
```

```
<?import javafx.scene.layout.*?>

<fx:root prefHeight="120.0" prefWidth="120.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Label layoutX="10.0" layoutY="35.0" text="IN1">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="10.0" layoutY="65.0" text="IN2">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="33.0" text="XOR" />
        <Line endY="120.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="120.0" startX="120.0" startY="8.0"
strokeWidth="3.0" />
        <Line endX="120.0" endY="120.0" startY="120.0" strokeWidth="3.0" />
        <Line endX="30.0" endY="8.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="8.0" startX="62.0" startY="8.0"
strokeWidth="3.0" />
        <TextField fx:id="in2TF" disable="false" layoutX="40.0" layoutY="58.0"
prefWidth="75.0" />
        <TextField fx:id="in1TF" disable="false" layoutX="40.0" layoutY="28.0"
prefWidth="75.0" />
        <TextField fx:id="destTF" disable="false" layoutX="40.0" layoutY="87.0"
prefWidth="75.0" />
        <Label layoutX="10.0" layoutY="93.0" text="DEST">
            <font>
                <Font size="9.0" />
            </font>
        </Label>
    </children>
</fx:root>
```

Add.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="120.0" prefWidth="120.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Label layoutX="10.0" layoutY="35.0" text="IN1">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="10.0" layoutY="65.0" text="IN2">
```

```

<font>
    <Font size="9.0" />
</font></Label>
<Label layoutX="33.0" text="ADD" />
<Line endY="120.0" startY="8.0" strokeWidth="3.0" />
<Line endX="120.0" endY="120.0" startX="120.0" startY="8.0"
strokeWidth="3.0" />
<Line endX="120.0" endY="120.0" startY="120.0" strokeWidth="3.0" />
<Line endX="30.0" endY="8.0" startY="8.0" strokeWidth="3.0" />
<Line endX="120.0" endY="8.0" startX="62.0" startY="8.0"
strokeWidth="3.0" />
<TextField fx:id="in2TF" disable="false" layoutX="40.0" layoutY="58.0"
prefWidth="75.0" />
<TextField fx:id="in1TF" disable="false" layoutX="40.0" layoutY="28.0"
prefWidth="75.0" />
<TextField fx:id="destTF" disable="false" layoutX="40.0" layoutY="87.0"
prefWidth="75.0" />
<Label layoutX="10.0" layoutY="93.0" text="DEST">
    <font>
        <Font size="9.0" />
    </font>
</Label>
</children>
</fx:root>

```

Div.fxml

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="120.0" prefWidth="120.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Label layoutX="10.0" layoutY="35.0" text="IN1">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="10.0" layoutY="65.0" text="IN2">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="33.0" text="DIV" />
        <Line endY="120.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="120.0" startX="120.0" startY="8.0"
strokeWidth="3.0" />
        <Line endX="120.0" endY="120.0" startY="120.0" strokeWidth="3.0" />
        <Line endX="30.0" endY="8.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="8.0" startX="62.0" startY="8.0"
strokeWidth="3.0" />
    </children>

```

```

        <Line endX="120.0" endY="8.0" startX="62.0" startY="8.0"
strokeWidth="3.0" />
        <TextField fx:id="in2TF" disable="false" layoutX="40.0" layoutY="58.0"
prefWidth="75.0" />
        <TextField fx:id="in1TF" disable="false" layoutX="40.0" layoutY="28.0"
prefWidth="75.0" />
        <TextField fx:id="destTF" disable="false" layoutX="40.0" layoutY="87.0"
prefWidth="75.0" />
        <Label layoutX="10.0" layoutY="93.0" text="DEST">
            <font>
                <Font size="9.0" />
            </font>
        </Label>
    </children>
</fx:root>

```

Mul.fxml

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="120.0" prefWidth="120.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Label layoutX="10.0" layoutY="35.0" text="IN1">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="10.0" layoutY="65.0" text="IN2">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="33.0" text="MUL" />
        <Line endY="120.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="120.0" startX="120.0" startY="8.0"
strokeWidth="3.0" />
        <Line endX="120.0" endY="120.0" startY="120.0" strokeWidth="3.0" />
        <Line endX="30.0" endY="8.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="8.0" startX="62.0" startY="8.0"
strokeWidth="3.0" />
        <TextField fx:id="in2TF" disable="false" layoutX="40.0" layoutY="58.0"
prefWidth="75.0" />
        <TextField fx:id="in1TF" disable="false" layoutX="40.0" layoutY="28.0"
prefWidth="75.0" />
        <TextField fx:id="destTF" disable="false" layoutX="40.0" layoutY="87.0"
prefWidth="75.0" />
        <Label layoutX="10.0" layoutY="93.0" text="DEST">
            <font>
                <Font size="9.0" />
            </font>
        </Label>
    </children>
</fx:root>

```

```
</font>
</Label>
</children>
</fx:root>
```

Sub.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="120.0" prefWidth="120.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Label layoutX="10.0" layoutY="35.0" text="IN1">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="10.0" layoutY="65.0" text="IN2">
            <font>
                <Font size="9.0" />
            </font></Label>
        <Label layoutX="33.0" text="SUB" />
        <Line endY="120.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="120.0" startX="120.0" startY="8.0"
strokeWidth="3.0" />
        <Line endX="120.0" endY="120.0" startY="120.0" strokeWidth="3.0" />
        <Line endX="30.0" endY="8.0" startX="33.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="8.0" startX="62.0" startY="8.0"
strokeWidth="3.0" />
        <TextField fx:id="in2TF" disable="false" layoutX="40.0" layoutY="58.0"
prefWidth="75.0" />
        <TextField fx:id="in1TF" disable="false" layoutX="40.0" layoutY="28.0"
prefWidth="75.0" />
        <TextField fx:id="destTF" disable="false" layoutX="40.0" layoutY="87.0"
prefWidth="75.0" />
        <Label layoutX="10.0" layoutY="93.0" text="DEST">
            <font>
                <Font size="9.0" />
            </font>
        </Label>
    </children>
</fx:root>
```

Ote.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.shape.*?>
```

```

<?import javafx.scene.image.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="64.0" prefWidth="64.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Label fx:id="tagName" layoutY="2.0" text="Tag_Name" />
        <Rectangle fx:id="dragOverIndicator" arcHeight="5.0" arcWidth="5.0"
fill="WHITE" height="20.0" layoutX="7.0" layoutY="44.0" stroke="BLACK"
strokeType="INSIDE" strokeWidth="0.0" width="50.0" />
        <Line fx:id="left_line" endX="18.0" endY="32.0" startY="32.0"
strokeWidth="2.0" />
        <Line fx:id="right_line" endX="42.0" endY="32.0" startX="64.0"
startY="32.0" strokeWidth="2.0" />
        <Arc fill="#1f93ff00" layoutX="30.0" layoutY="32.0" length="160.0"
radiusX="12.0" radiusY="12.0" startAngle="100.0" stroke="BLACK"
strokeType="INSIDE" strokeWidth="2.0" />
        <Arc fill="#1f93ff00" layoutX="31.0" layoutY="32.0" length="160.0"
radiusX="12.0" radiusY="12.0" startAngle="280.0" stroke="BLACK"
strokeType="INSIDE" strokeWidth="2.0" />
    </children>
</fx:root>

```

Otl.fxml

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import javafx.scene.image.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="64.0" prefWidth="64.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Label fx:id="tagName" layoutY="2.0" text="Tag_Name" />
        <Rectangle fx:id="dragOverIndicator" arcHeight="5.0" arcWidth="5.0"
fill="WHITE" height="20.0" layoutX="7.0" layoutY="44.0" stroke="BLACK"
strokeType="INSIDE" strokeWidth="0.0" width="50.0" />
        <Line fx:id="left_line" endX="18.0" endY="32.0" startY="32.0"
strokeWidth="2.0" />
        <Line fx:id="right_line" endX="42.0" endY="32.0" startX="64.0"
startY="32.0" strokeWidth="2.0" />
        <Arc fill="#1f93ff00" layoutX="30.0" layoutY="32.0" length="160.0"
radiusX="12.0" radiusY="12.0" startAngle="100.0" stroke="BLACK"
strokeType="INSIDE" strokeWidth="2.0" />
    </children>
</fx:root>

```

```
<Arc fill="#1f93ff00" layoutX="31.0" layoutY="32.0" length="160.0"
radiusX="12.0" radiusY="12.0" startAngle="280.0" stroke="BLACK"
strokeType="INSIDE" strokeWidth="2.0" />
<Text layoutX="27.0" layoutY="37.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="L">
    <font>
        <Font name="System Bold" size="14.0" />
    </font>
</Text>
</children>
</fx:root>
```

Otu.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import javafx.scene.image.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="64.0" prefWidth="64.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Label fx:id="tagName" layoutY="2.0" text="Tag_Name" />
        <Rectangle fx:id="dragOverIndicator" arcHeight="5.0" arcWidth="5.0"
fill="WHITE" height="20.0" layoutX="7.0" layoutY="44.0" stroke="BLACK"
strokeType="INSIDE" strokeWidth="0.0" width="50.0" />
        <Line fx:id="left_line" endX="18.0" endY="32.0" startY="32.0"
strokeWidth="2.0" />
        <Line fx:id="right_line" endX="42.0" endY="32.0" startX="64.0"
startY="32.0" strokeWidth="2.0" />
        <Arc fill="#1f93ff00" layoutX="30.0" layoutY="32.0" length="160.0"
radiusX="12.0" radiusY="12.0" startAngle="100.0" stroke="BLACK"
strokeType="INSIDE" strokeWidth="2.0" />
        <Arc fill="#1f93ff00" layoutX="31.0" layoutY="32.0" length="160.0"
radiusX="12.0" radiusY="12.0" startAngle="280.0" stroke="BLACK"
strokeType="INSIDE" strokeWidth="2.0" />
        <Text layoutX="25.0" layoutY="38.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="U">
            <font>
                <Font name="System Bold" size="14.0" />
            </font>
</Text>
</children>
</fx:root>
```

Xic.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.shape.*?>
<?import javafx.scene.image.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="64.0" prefWidth="64.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Label fx:id="tagName" layoutY="2.0" text="Tag_Name" />
        <Rectangle fx:id="dragOverIndicator" arcHeight="5.0" arcWidth="5.0"
fill="WHITE" height="20.0" layoutX="7.0" layoutY="44.0" stroke="BLACK"
strokeType="INSIDE" strokeWidth="0.0" width="50.0" />
        <Line fx:id="left_line" endX="28.0" endY="32.0" startY="32.0"
strokeWidth="2.0" />
        <Line fx:id="right_line" endX="36.0" endY="32.0" startX="64.0"
startY="32.0" strokeWidth="2.0" />
        <Line fx:id="left_bar" endX="28.0" endY="42.0" startX="28.0"
startY="22.0" strokeWidth="2.0" />
        <Line fx:id="right_bar" endX="36.0" endY="42.0" startX="36.0"
startY="22.0" strokeWidth="2.0" />
        <Line fx:id="center_bar" endX="30.0" endY="42.0" startX="34.0"
startY="22.0" />
    </children>
</fx:root>
```

Xio.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.shape.*?>
<?import javafx.scene.image.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="64.0" prefWidth="64.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1" >
    <children>
        <Label fx:id="tagName" layoutY="2.0" text="Tag_Name" />
        <Rectangle fx:id="dragOverIndicator" arcHeight="5.0" arcWidth="5.0"
fill="WHITE" height="20.0" layoutX="7.0" layoutY="44.0" stroke="BLACK"
strokeType="INSIDE" strokeWidth="0.0" width="50.0" />
        <Line fx:id="left_line" endX="28.0" endY="32.0" startY="32.0"
strokeWidth="2.0" />
        <Line fx:id="right_line" endX="36.0" endY="32.0" startX="64.0"
startY="32.0" strokeWidth="2.0" />
        <Line fx:id="left_bar" endX="28.0" endY="42.0" startX="28.0"
startY="22.0" strokeWidth="2.0" />
        <Line fx:id="right_bar" endX="36.0" endY="42.0" startX="36.0"
startY="22.0" strokeWidth="2.0" />
        <Line fx:id="center_bar" endX="30.0" endY="42.0" startX="34.0"
startY="22.0" />
    </children>
</fx:root>
```

```
<Line fx:id="right_bar" endX="36.0" endY="42.0" startX="36.0"
startY="22.0" strokeWidth="2.0" />
</children>
</fx:root>
```

Ctd.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="90.0" prefWidth="120.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1">
<children>
    <Label layoutX="3.0" layoutY="40.0" text="Counter">
        <font>
            <Font size="9.0" />
        </font></Label>
    <Label layoutX="3.0" layoutY="70.0" text="Count">
        <font>
            <Font size="9.0" />
        </font></Label>
    <Label layoutX="33.0" text="CTD" />
    <Line endY="90.0" startY="8.0" strokeWidth="3.0" />
    <Line endX="120.0" endY="90.0" startX="120.0" startY="8.0"
strokeWidth="3.0" />
    <Line endX="120.0" endY="90.0" startY="90.0" strokeWidth="3.0" />
    <Line endX="30.0" endY="8.0" startY="8.0" strokeWidth="3.0" />
    <Line endX="120.0" endY="8.0" startX="62.0" startY="8.0"
strokeWidth="3.0" />
    <ComboBox fx:id="timerCB" layoutX="37.0" layoutY="33.0"
prefWidth="80.0" />
    <TextField fx:id="countTF" disable="true" layoutX="40.0" layoutY="62.0"
prefWidth="75.0" />
</children>
</fx:root>
```

Ctu.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="90.0" prefWidth="120.0"
type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1">
```

```

<children>
    <Label layoutX="3.0" layoutY="40.0" text="Counter">
        <font>
            <Font size="9.0" />
        </font></Label>
    <Label layoutX="3.0" layoutY="70.0" text="Count">
        <font>
            <Font size="9.0" />
        </font></Label>
    <Label layoutX="33.0" text="CTU" />
    <Line endY="90.0" startY="8.0" strokeWidth="3.0" />
    <Line endX="120.0" endY="90.0" startX="120.0" startY="8.0" strokeWidth="3.0" />
    <Line endX="120.0" endY="90.0" startY="90.0" strokeWidth="3.0" />
    <Line endX="30.0" endY="8.0" startY="8.0" strokeWidth="3.0" />
    <Line endX="120.0" endY="8.0" startX="62.0" startY="8.0" strokeWidth="3.0" />
    <ComboBox fx:id="timerCB" layoutX="37.0" layoutY="33.0" prefWidth="80.0" />
    <TextField fx:id="countTF" disable="true" layoutX="40.0" layoutY="62.0" prefWidth="75.0" />
</children>
</fx:root>

```

Ton.fxml

```

<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<fx:root prefHeight="90.0" prefWidth="120.0" type="javafx.scene.layout.AnchorPane" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Label layoutX="5.0" layoutY="40.0" text="Timer" />
        <Label layoutX="5.0" layoutY="70.0" text="Count" />
        <Label layoutX="33.0" text="TON" />
        <Line endY="90.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="90.0" startX="120.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="90.0" startY="90.0" strokeWidth="3.0" />
        <Line endX="30.0" endY="8.0" startY="8.0" strokeWidth="3.0" />
        <Line endX="120.0" endY="8.0" startX="62.0" startY="8.0" strokeWidth="3.0" />
        <ComboBox fx:id="timerCB" layoutX="37.0" layoutY="33.0" prefWidth="80.0" />
        <TextField fx:id="countTF" disable="true" layoutX="40.0" layoutY="62.0" prefWidth="75.0" />
    </children>
</fx:root>

```

Appendix K: Project name compilation files

Project name.id file contents

```
XIC: EN(0x102, BitNumber:0), ENO(0x102, BitNumber:1), DI0(0x070,  
BitNumber:0),  
ADD: EN(0x103, BitNumber:0), ENO(0x103, BitNumber:1), AI0(0x060), AI1(0x062),  
tag1(0x101),  
EQU: EN(0x104, BitNumber:0), ENO(0x104, BitNumber:1), AI2(0x064),  
tag1(0x101),  
OTE: EN(0x105, BitNumber:0), ENO(0x105, BitNumber:1), DO0(0x073,  
BitNumber:0),
```

Project name.asm file contents

```
org 0x2000  
0x2000 : MOVLB 0x1  
0x2002 : BSF 0x102, 0, 1  
0x2004 : MOVLB 0x0  
0x2006 : BTFSC 0x70, 0, 1  
0x2008 : BRA 0x003  
0x200A : MOVLB 0x1  
0x200C : BSF 0x102, 1, 1  
0x200E : BRA 0x002  
0x2010 : MOVLB 0x1  
0x2012 : BCF 0x102, 1, 1  
0x2014 : MOVLB 0x1  
0x2016 : BTFSS 0x102, 1, 1  
0x2018 : BRA 0x003  
0x201A : MOVLB 0x1  
0x201C : BSF 0x103, 0, 1  
0x201E : BRA 0x003  
0x2020 : MOVLB 0x1  
0x2022 : BCF 0x103, 0, 1  
0x2024 : BRA 0x007  
0x2026 : BSF 0x103, 1, 1  
0x2028 : MOVLB 0x0  
0x202A : MOVF 0x60, 0, 1  
0x202C : MOVLB 0x0  
0x202E : ADDWF 0x62, 0, 1  
0x2030 : MOVLB 0x1  
0x2032 : MOVWF 0x101, 1  
0x2034 : MOVLB 0x1  
0x2036 : BSF 0x104, 0, 1  
0x2038 : BTFSS 0x104, 0, 1  
0x203A : GOTO 0x080F4  
0x203E : MOVLB 0x0  
0x2040 : MOVF 0x65, 0, 1  
0x2042 : SUBLW 0x00  
0x2044 : MOVLB 0xF  
0x2046 : BTFSC 0xFD8, 2, 1  
0x2048 : GOTO 0x04078  
0x204C : MOVLB 0x1  
0x204E : MOVF 0x101, 0, 1  
0x2050 : MOVLB 0x0
```

```
0x2052 : SUBWF 0x64, 0, 1
0x2054 : MOVLB 0xF
0x2056 : BTFSC 0xFD8, 2, 1
0x2058 : GOTO 0x04078
0x205C : MOVLB 0x1
0x205E : BSF 0x104, 1, 0
0x2060 : GOTO 0x080F4
0x2064 : MOVLB 0x1
0x2066 : BCF 0x104, 1, 0
0x2068 : NOP
0x206A : MOVLB 0x1
0x206C : BTFSS 0x104, 1, 1
0x206E : BRA 0x003
0x2070 : MOVLB 0x1
0x2072 : BSF 0x105, 0, 1
0x2074 : BRA 0x003
0x2076 : MOVLB 0x1
0x2078 : BCF 0x105, 0, 1
0x207A : BRA 0x004
0x207C : BSF 0x105, 1, 1
0x207E : MOVLB 0x0
0x2080 : BSF 0x73, 0, 1
0x2082 : BRA 0x004
0x2084 : MOVLB 0x0
0x2086 : BCF 0x73, 0, 1
0x2088 : MOVLB 0x1
0x208A : BCF 0x105, 1, 1
RET
```

Project name.pgm file contents

```
0x2000 : 0x01
0x2001 : 0x01
0x2002 : 0x02
0x2003 : 0x81
0x2004 : 0x00
0x2005 : 0x01
0x2006 : 0x70
0x2007 : 0xB1
0x2008 : 0x03
0x2009 : 0xD0
0x200A : 0x01
0x200B : 0x01
0x200C : 0x02
0x200D : 0x83
0x200E : 0x02
0x200F : 0xD0
0x2010 : 0x01
0x2011 : 0x01
0x2012 : 0x02
0x2013 : 0x93
0x2014 : 0x01
0x2015 : 0x01
0x2016 : 0x02
0x2017 : 0xA3
0x2018 : 0x03
```

```
0x2019 : 0xD0
0x201A : 0x01
0x201B : 0x01
0x201C : 0x03
0x201D : 0x81
0x201E : 0x03
0x201F : 0xD0
0x2020 : 0x01
0x2021 : 0x01
0x2022 : 0x03
0x2023 : 0x91
0x2024 : 0x07
0x2025 : 0xD0
0x2026 : 0x03
0x2027 : 0x83
0x2028 : 0x00
0x2029 : 0x01
0x202A : 0x60
0x202B : 0x51
0x202C : 0x00
0x202D : 0x01
0x202E : 0x62
0x202F : 0x25
0x2030 : 0x01
0x2031 : 0x01
0x2032 : 0x01
0x2033 : 0x6F
0x2034 : 0x01
0x2035 : 0x01
0x2036 : 0x04
0x2037 : 0x81
0x2038 : 0x04
0x2039 : 0xA1
0x203A : 0xF4
0x203B : 0xEF
0x203C : 0x80
0x203D : 0xFF
0x203E : 0x00
0x203F : 0x01
0x2040 : 0x65
0x2041 : 0x51
0x2042 : 0x00
0x2043 : 0x08
0x2044 : 0x0F
0x2045 : 0x01
0x2046 : 0xD8
0x2047 : 0xB5
0x2048 : 0x78
0x2049 : 0xEF
0x204A : 0x40
0x204B : 0x00
0x204C : 0x01
0x204D : 0x01
0x204E : 0x01
0x204F : 0x51
0x2050 : 0x00
0x2051 : 0x01
```

```
0x2052 : 0x64
0x2053 : 0x5D
0x2054 : 0x0F
0x2055 : 0x01
0x2056 : 0xD8
0x2057 : 0xB5
0x2058 : 0x78
0x2059 : 0xEF
0x205A : 0x40
0x205B : 0x00
0x205C : 0x01
0x205D : 0x01
0x205E : 0x04
0x205F : 0x82
0x2060 : 0xF4
0x2061 : 0xEF
0x2062 : 0x80
0x2063 : 0xFF
0x2064 : 0x01
0x2065 : 0x01
0x2066 : 0x04
0x2067 : 0x92
0x2068 : 0x00
0x2069 : 0x00
0x206A : 0x01
0x206B : 0x01
0x206C : 0x04
0x206D : 0xA3
0x206E : 0x03
0x206F : 0xD0
0x2070 : 0x01
0x2071 : 0x01
0x2072 : 0x05
0x2073 : 0x81
0x2074 : 0x03
0x2075 : 0xD0
0x2076 : 0x01
0x2077 : 0x01
0x2078 : 0x05
0x2079 : 0x91
0x207A : 0x04
0x207B : 0xD0
0x207C : 0x05
0x207D : 0x83
0x207E : 0x00
0x207F : 0x01
0x2080 : 0x73
0x2081 : 0x81
0x2082 : 0x04
0x2083 : 0xD0
0x2084 : 0x00
0x2085 : 0x01
0x2086 : 0x73
0x2087 : 0x91
0x2088 : 0x01
0x2089 : 0x01
0x208A : 0x05
```

0x208B : 0x93
0x208C : 0x12
0x208D : 0x00

Project name.hex file contents

```
:101D16001AEC0FF0B886B06ACF0EAF6EAC8413DD46
:101D2600020E0B6E040E0C6EBA0E0D6E0D2EFED745
:101D36000C2EFCD70B2EFAD700000F0EC112070E81
:101D4600B4129280516A94909494816A936A8290B4
:101D56008294736B706B7F6B78DF010EE56F1DEC01
:101D660009F00C0EE56F1DEC09F080A002D0F3DE41
:081D760001D012DEFAD7FFD7FD
:101E340010EE29F0270E006E010E016EA70EF66E4D
:101E44001D0EF76E000EF86ED2EC0BF010EE80F063
:101E5400110E006E010E016E010EF66E1E0EF76E6F
:0A1E6400000EF86ED2EC0BF0120035
:101D7E00090229000101008032090400000203005B
:101D8E000000092101010001222100070581034005
:091D9E00000107050103400001EA
:091DA70052554E204D4F444500F9
:0D1DB00050524F4752414D204D4F444500C9
:101DBD0050524F4752414D4D494E472E2E2E2EED
:011DCD000015
:101DCE000600FF0901A10119012940150026FF0097
:101DDE0075089540810219012940750895409102B8
:011DEE00C034
:101DEF00120100020000000834120100010001027C
:021DFF000001E1
:101E0100434F4E4E454354454420544F2050432E9A
:011E110000D0
:101E120010035600540020004C0061006200730061
:0E1E22000E0356005400200050004C004300F8
:041E3000040309049A
:1020000001010281000170b103d00101028302d0fd
:1020100001010293010102a303d00101038103d056
:102020000101039107d00383000160510001622583
:102030000101016f0101048104a1f4ef80ff00019f
:10204000655100080f01d8b578ef4000010101513a
:102050000001645d0f01d8b578ef400001010482f2
:10206000f4ef80ff010104920000010104a303d0fa
:102070000101058103d00101059104d00583000110
:0e208000738104d000017391010105931200d9
:020000040030CA
:0E000000210C391EFF8081FF0FC00FE00F4062
:00000001FF
```