DESIGN OF AN ADAPTABLE LADDER EDITOR FOR PROGRAMMABLE LOGIC CONTROLLER

Assmaa A. Fahad and Omar R. Abed Al-Aziz
College of Science, Computer Science Department, Baghdad University.
E_mail: assmaa_fahad@yahoo.com

Abstract

Ladder diagrams, which have been used for decades for describing relay circuits, are now being utilized for programming Programmable Logic Controllers (PLC). This is so because many practicing engineers and technicians are familiar with these diagrams, and fell comfortable working with them.

The aim of this work is to design and implement a visual editor for ladder programming language that helps the programmer to write Ladder programs using PC environment, and when this editor connected with a real-time database it can be used as a ladder man machine interface to monitor a factory from outside field. This approach provides a good man machine interface than programming with mnemonic logic instruction.

Normally the ladder programs have a very big size, so the editor has to find the most suitable method to store these programs and to navigate between the pages of these programs in a very fast response time.

The designed editor is written using Visual Basic programming language and it is tested to write application programs for different industrial control processes and shows a very accepted results.

Keywords: Programmable logic Controller, Ladder Program, PLC Editor

Introduction

The Programmable Logic Controller (PLC) founds in industry has evolved from the needs for a control system that can be easily reprogrammed as changes occur or as new products develop. The automotive industry is faced annually with major changes in production as new models are designed. This changeover requires electricians and maintenance personnel to put in long hours to rewrite relay type controls. Each changeover period was costly to the industry, and it often forced changes to be infrequent and as simple as possible. As a result of these difficulties, the PLC was developed to simplify the problem of changing control system periodically [4,1].

The PLC is a solid state device used to control machine or process operation by means of a stored program and feedback of data from input/output devices. It is composed primarily of three parts: PLC processor module, the Input/Output modules and the programming device [6,5].

The PLC processor module reads input data from various sensing devices, executes the stored user program from memory, and sends appropriate output commands to control devices. This process of reading inputs, executing the program, and controlling output is done on a continuous basis called scanning [4,1].

PLC Operation Modes

Depending on the PLC system, there are different types of operating modes. The common operating modes are PROGRAM mode, RUN mode, and TEST mode [2].

With PROGRAM mode the processor module prepares for receiving/ transmitting data from/to the programming device with the PLC memory.

In RUN mode the PLC system begins its scanning operation by looking for input signals from input devices, execute the user program and according to the user logic program sends output signals to the output devices.

The third mode of operation is the TEST mode. In this mode the PLC system operates just as in RUN mode except that the output modules remain in the OFF state. This mode of operation is normally used to

test the written PLC program before implementing it, the user activates input devices and watches the operation of the program without wary about something that might moves to the wrong state.

PLC Programming Languages

The term PLC programming language refers to the method by which the user communicates information to the PLC. There are four types of languages normally encountered in programmable controllers: Ladder diagram, Boolean mnemonics, Function blocks, and High level literal language.

Ladder diagram and Boolean mnemonics form basic PLC languages, while function blocks, and High level literal language are considered high level languages [2].

The basic PLC languages consists of a set of instructions that perform the most primitive type of control functions: relay replacement, timing and counting operation. The high level languages have been brought about the need to execute more powerful instructions that go beyond simple timing, counting and on/off control. High level languages are suited for operations such as analog control, data manipulation, reporting, and other functions that are not possible with basic instruction sets.

The main requirement from any PLC programming language is that it may be easily understood and used in a control situation. This implies the need for a language type that provides commands very closed to the function required by the control engineer, but without complexity and learning time associated with most computer languages. For this reason ladder diagram language is chosen, because it has been the most common method of describing relay logic circuit, so it is only natural to base PLC programming on them in order to create a familiar environment for the user and designer of logic control systems.

Ladder Diagram Programming Language

Ladder diagram is the most popular symbolic language. This language is a natural choice for PLCs since it was already used to represent electromechanical relays, counters, and timers which PLCs were replacing.

The ladder diagram language is used to create a PLC user logic program. It is almost composed of six groups of instructions: relay-type, timer/counter, arithmetic instruction, data manipulation, data transfer, and program control [2]. These instructions are formatted to obtain the desired control logic that is to be entered into the PLC memory.

The main function of the ladder diagram program is to control outputs based on input conditions. This control is accomplished through the use of what is referred to as a ladder rung.

Each ladder diagram has two vertical lines, the left vertical is considered connected to the supply voltage, while the right one is grounded as shown in Fig-1. Thus, each horizontal line, or ladder rung, represents a separate circuit performing its own task. The basic ladder diagram element is the relay, each relay consists of a coil and a number of contacts, some normally open (**-[ ]-**), and some normally closed (**-[/]-**). When a coil is energized, the contacts belonging to the same relay switch (i.e. with the same identifying number), a normally open contact closes, while a normally closed one opens.

The right most PLC symbol within a ladder rung is referred to as an output instruction (**–( )-**). This instruction generates some sort of response based on True/False status of the ladder rung. This instruction is almost always a relay coil or similar instruction. To the left of the output instruction there are the input conditions represented by contact instructions for the ladder rung.

Each contact or coil symbol is referenced with an address number that

indicate what PLC input is connected to what input device, and what PLC output will drive what output device. The addressing of real inputs and outputs, as well as internals, depends upon the PLC model used instruction addressing format.

The Designed Ladder Editor

Since the Ladder language is a symbolic PLC language, a software development system is needed to provide graphic programming. This software is considered as a Man Machine Interface (MMI) that provides the user with the essential graphic tools to draw relay ladder diagram to program the PLC [3]. The designed editor consists of two main sections: Editing section and PLC File Construction section.

A-The Editing Section

The editing section provides a graphic editing facilities that allow the user to enter the desired ladder diagram symbols. It is designed to consist of two areas: the graphic screen area and the menu area, as shown in Fig-2.

The graphic screen area is used to enter relay ladder symbols in order to construct each rung of the ladder program. This area is designed to consist of a net of points to simplify fixing the ladder symbol either vertically or horizontally. The symbols bar in this area is provided to display some of the ladder symbols that can be used to build the ladder program, and two text fields, labeled with *logic and text*, used to specify the symbol name and the symbol address that helps to understand and maintain the ladder program. The symbols are added one symbol at a time and each new created symbol must be connected with either input or output symbol to construct the ladder program.

The menu area displays a list of commands, File, View, and Tools. The File menu is used to create and save a file using the options provided in this menu such as New, Open, Save, and SaveAs. The New option is used to create new instance and once the user finished working on the file, the Save option allows the user to save a copy of the finished ladder program. SaveAs option is also provided to allow building different versions of the written ladder program.

The View menu provides different options that allow the user to change the configuration of the ladder editor. Some of these options are used to specify the maximum number of pages needed to construct the ladder program, the number of rungs in each page, and to define the maximum number of symbols each page can hold. Each page can be configured to hold up to 7*15 symbols.

The Tools menu provides a flexibility of adding new ladder symbols to the editor or deleting unused symbols from the editor (see Fig-3).

The editor also provides many editing facilities such as online documentation help, navigation between the program pages, editing facilities (Cut-Copy-Paste-Undo) and clear function that can be used to clear the contents of the current page in one command.

B- PLC File Construction Section

The PLC file construction section is responsible for loading and saving the PLC ladder program. The ladder editor allows the user to enter the desired ladder diagram symbols one symbol at a time. The entered symbols are displayed on the graphic screen area and an information record about this symbol is stored in a graphic data table. The graphic data table consists of many records of information, one record for each symbol in the rung. The contents of this table will be saved in a text file with a necessary header information about each ladder program that helps the editor to reload the program. These header information consists of the configuration information in addition to any remark statements in the program inserted by the user.

The graphic data table is stored in the memory using dynamic array data structure. Because of the large size of ladder program, the information stored in a graphic data table need to have as minimum size as possible. Therefore, the record is designed to hold the following information about each symbol:

X Position	Y Position	Symbol Code	Page NO.	Logic State	Text

The program symbols are stored as a symbol code rather than the symbol itself, therefore a data base file is used to store the symbols and their code numbers. The editor refer to this data base file during the save and load operations.

After a certain number of designed pages, depending on the number of symbols inserted on these pages, the editor save the contents of the dynamic array in a temporary file and free the located memory in order to reuse it. Saving the contents of the graphic data table in a temporary file will speed up the operations with this data structure because of his always small size. The contents of the temporary file is reloaded in the dynamic array when the user review the designed ladder pages.

The Ladder Editor as a Man Machine Interface

During RUN and TEST modes of operations, the operator needs to monitor the operation of the plant during the execution of the PLC program through MMI. All objects in the designed ladder program are linked to a real-time point(field address) in order to reflect its status to the objects in the program. The MMI needs a real time data collected from the field of the plant and to record the initial states of the components, which is ON or OFF, in a data base file. During the scanning operation the editor will check the new status with the saved one, and with any change the editor will reflect this new status to the related object in the ladder program, and updates the data base file accordingly.

During TEST mode the user can monitor the behavior of the plant during executing the ladder program and with any illegal operation, the operator can change the program to overcome any illegal operation in the program before implementing it.

In RUN mode the MMI option helps the operator to monitor the plant operations and with any fault during the work time the operator can easily discover the fault reason and solve the problem in a very fast and easy way.

Real Time Application Examples Using the Designed Ladder Editor

A number of different real-time application examples are used to test the designed editor. The selected examples are chooses to need different requirements from the editor in order to test the facilities provided by the editor.

One of the application program examples is the Neutralization system shown in Fig-4. In this example a certain amount of solution is added to a tank; heated and chemically treated and then sent out to the next tank. The sensors **ts** and **as** indicate whether or not the solution has the correct temperature and PH, respectively. When **ts** and **as** are both activated, the neutralization is complete. The level switches **ls1, ls2,** and **ls3** are activated whenever the level in the tank is at or above a given level [7].The neutralization process is to proceed as follows:

1. Initially, all the valves are closed, the mixer **m** and heater **h** are OFF, and the reaction tank is empty.
2. When the START button is pressed, open **v1** until **ls2** is activated. This fills the tank with the solution to be neutralized.
3. When the solution level rises above **ls2**, start the mixer **m**. When the level drops below **ls1**, stop the mixer.
4. Whenever the temperature of the solution is below a preset point, energize the heater **h**.
5. When the **PH** of the solution is unbalanced, open **v2** to add neutralizer.
6. If the tank becomes full, indicated by the activation of **ls3**, close **v2** to stop the inflow of neutralizer. Next, open **v4** to reduce the level of solution to the point indicated by **ls2**. Then close **v4** and process to step 5.
7. When both the temperature and **PH** of the solution are correct, de-energize the heater and close **v2**. Then open **v3** to drain the tank. When the tank is empty, indicated by the deactivation of **ls1**, close **v3** and proceed with step1.

In addition to the above control sequence, two indicator lights, **t1** and **a1**, are regulated. Lights **t1** and **a1** should turn ON whenever the solution level is above **ls2**; and

the temperature and PH have reached their preset values.

The designed ladder program and a part from PLC file for this application example is shown in Fig-5 and Fig-6 respectively. The PLC file is constructed as explained in section 4.2.

Discussions and Conclusions

The designed ladder programs are tested by implementing a real-time application program with a simulation data and it proves a good response time. Writing different control ladder programs for these examples proves that the designed PLC ladder editor contains the most ladder instructions that can be used for programming PLCs and it provides helpful functions that make easy creating, updating and manipulating of any application program.

The flexibility of the designed editor due to the different complicated data structure used in the programming operation make the load, save, and manipulate operations performed in a very fast and efficient way. The written ladder programs are easy to understand and maintain due to the inserted text facility provided by the editor with each ladder symbol.

Saving the contents of the graphic data table in a temporary file periodically, minimize the reserved memory space and this will speedup the response time of the program especially the navigation between the designed pages will be faster.

References

1. AL-ISSA, A. M., "Design and Implementation of Graphical User Interface for Industrial Applications", M.Sc. thesis, college of science, University of Baghdad, 2004.
2. AL-KHUDAIRY, T. F., "Design of a VMEbus-based for Programmable Logic Controller (PLC)", Microprocessors and Microsystems, Elsevier Science, Vol. 21, Page 329-336, 1998.
3. EBERTS, R.E. "User Interface Design", Prentice-Hall Inc., 1994.
4. JONES, C. T., and BRYAN, L. A., "Programmable Controllers Concepts and Applications", IPC/ASTEC, 1983.
5. KISSEL, T.E., "Understanding and using Programmable Controller", Prectice-Hall Inc., 2002.
6. PETRUZELLA, F. D., "Programmable LogicControllers", Mc-Graw-Hill Inc., 1989.
7. WARNOCK, I. G., "Programmable Controllers, Operation and Application", Prentice-Hall Inc., 1988.

المستخلص

استخدمت لغة البرمجة السلمية ولسنوات عديدة ولازالت تستخدم لبرمجة منظومات السيطرة المنطقية المبرمجة وذلك بسبب سهولة التعامل مع هذا النوع من اللغات من قبل المهندسين والفنيين العاملين في مجال برمجة منظومات السيطرة المنطقية المبرمجة. الهدف من هذا البحث هو تصميم وتنفيذ محرر مرئي لهذه اللغة البرمجية يسهل على المبرمج كتابة برامج السيطرة ويمكن استخدامه إذا ما ربط مع بيانات حقيقية كواجهة لمراقبة عمل منظومة معينة عن طريق الحاسبة من غرفة السيطرة حيث إن استخدام هذه اللغة في توفير واجهات المراقبة هي من انسب الطرق مقارنة بباقي لغات البرمجة الأخرى المتوفرة لبرمجة منظومات السيطرة

تمتاز البرامج المكتوبة باستخدام لغة البرمجة السلمية بكبر حجمها لذا يجب اختيار أفضل وانسب الطرق للتعامل مع هذا النوع من البرامج من حيث طريقة عرض البرامج وطريقة خزنها وأفضل الطرق للتنقل بين صفحات البرنامج الواحد وبأسرع وقت ممكن. تم استخدام لغة البرمجة فيجوال بيسك لكتابة برامج المحرر وقد تم فحص إمكانيات المحرر بكتابة البرامج التطبيقية لعدة أنظمة وقد حقق نتائج جيدة.

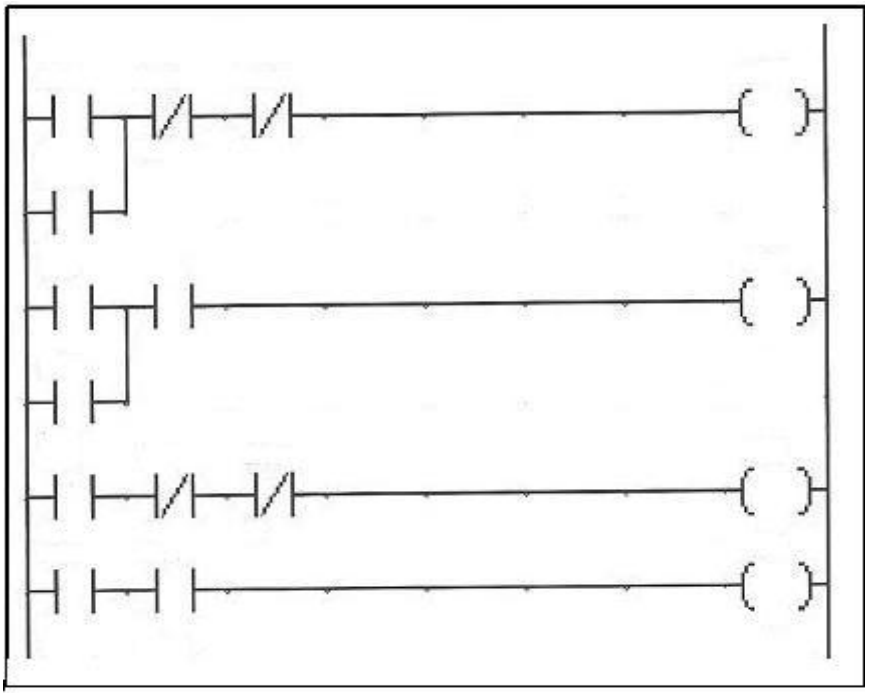

Fig-1 Ladder Program

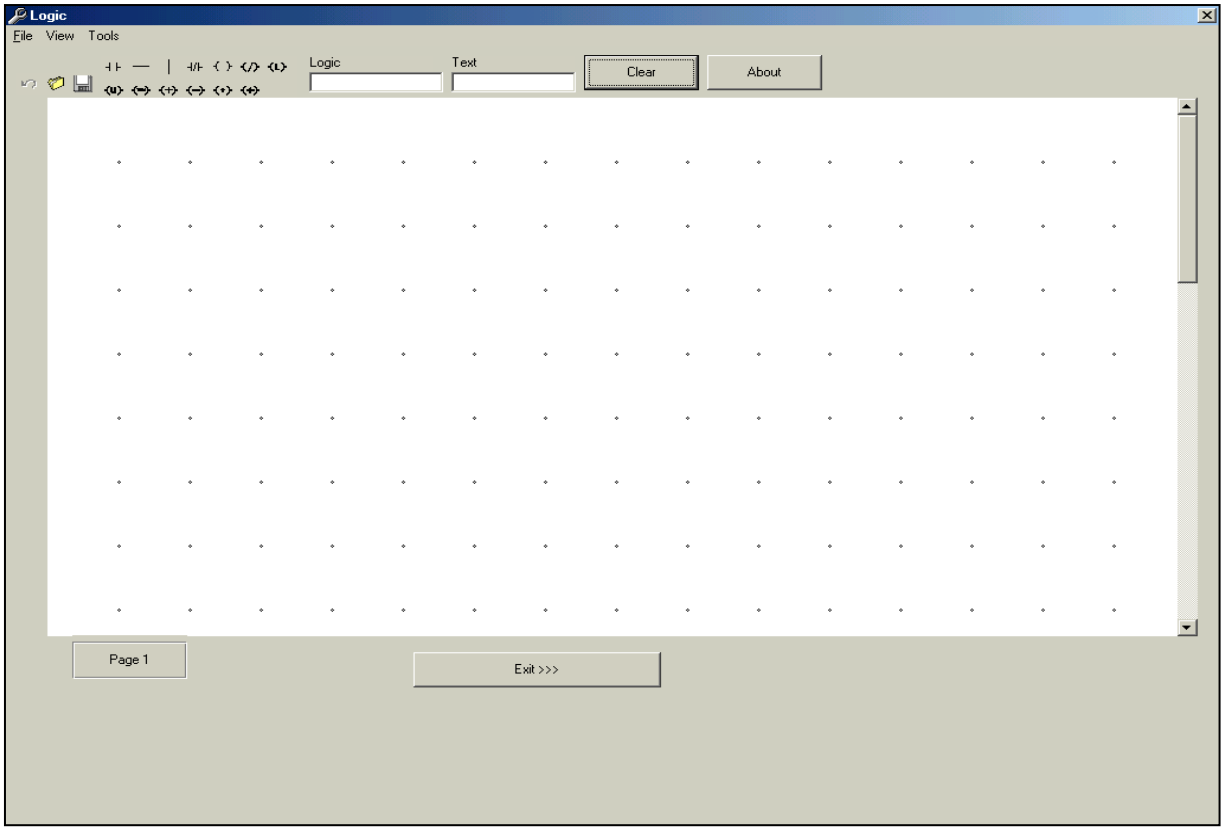

Fig-2 The Designed Ladder Editor

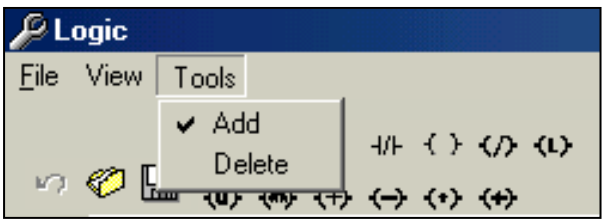

Fig-3 The Tools Menu Options

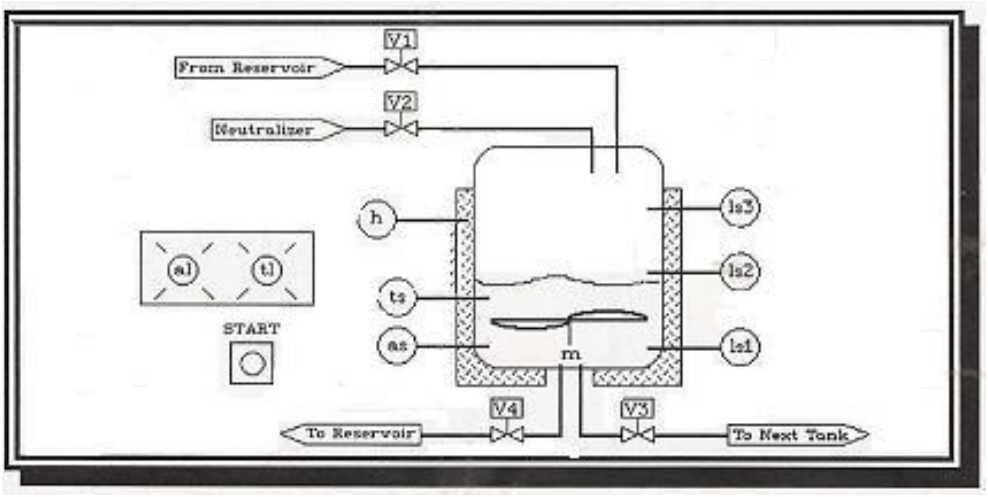

Fig-4 The Neutralization System

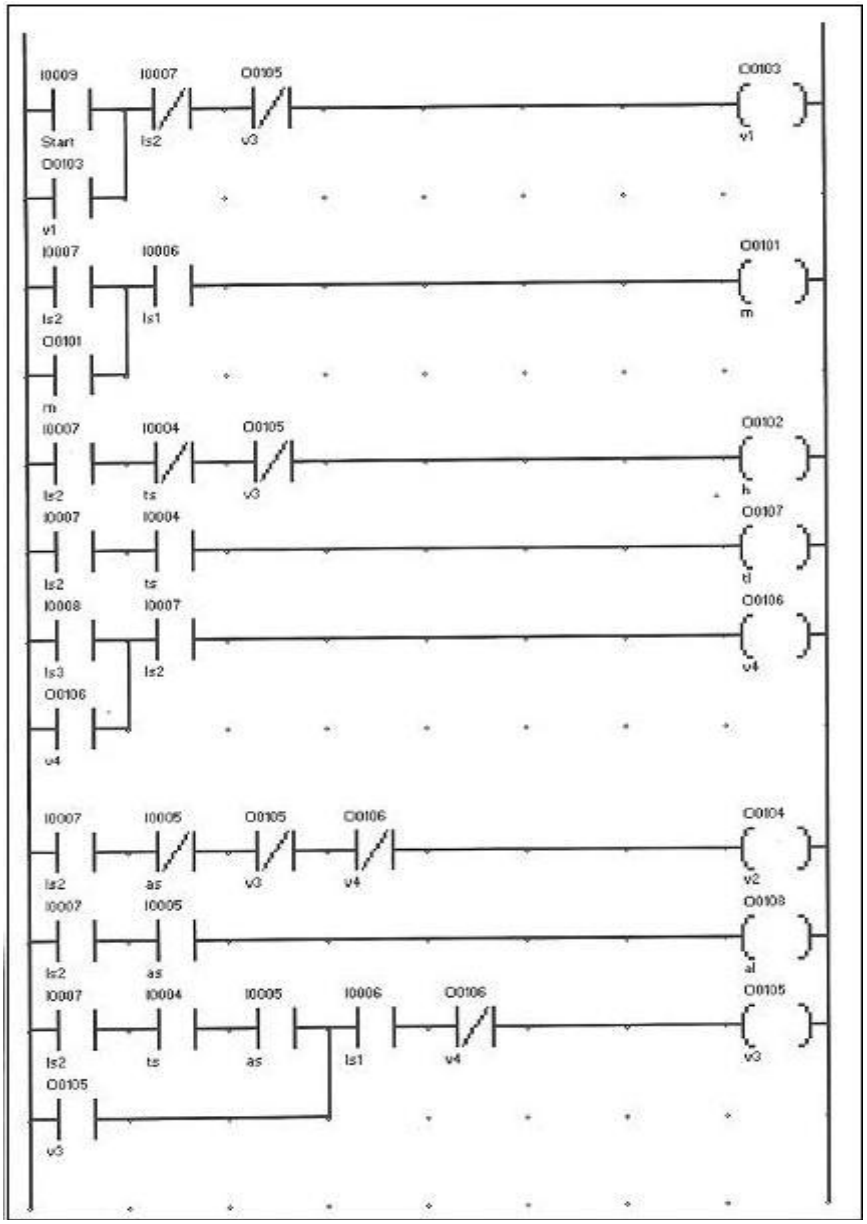

Fig-5 The Neutralization System Ladder Program

60 330 1 1 I007 ls2 60 30 1 2 I007 ls2 60 90 1 2 I007 ls3 60 150 1 2 I007 ls2 60 90 1 1 O0103 v1 60 210 1 1 O0101 m 60 450 1 1 O0106 v4 60 390 1 1 I008 ls3 60 210 1 2 O0105 v3 90 60 3 1 90 180 3 1 90 420 3 1 120 150 1 1 I0006 ls1 120 330 1 1 I0004 ts 120 150 1 2 I004 ts 120 390 1 1 I0007 ls2 120 90 1 2 I0005 as 180 150 1 2 I005 as 120 30 4 2 I0005 as 180 30 4 1 O0105 v3 180 30 4 2 O0105 v3240 30 4 2 O0106 v4 300 150 4 2 O0106 v4 240 150 1 2 I0006 ls1 120 210 2 2 180 210 2 2 210 180 3 2 60 330 1 1 I007 ls2 60 30 1 2 I007 ls2 60 90 1 2 I007 ls3 60 150 1 2 I007 ls2 60 90 1 1 O0103 v1 60 210 1 1 O0101 m 60 450 1 1 O0106 v4 60 390 1 1 I008 ls3 60 210 1 2 O0105 v3 90 60 3 1 90 180 3 1 90 420 3 1 120

Fig-6 A Part From Neutralization PLC File