LLM-Based Program Analysis for Ladder Program

Author: *Naoki Sugawara**

**Information Technology R&D Center*

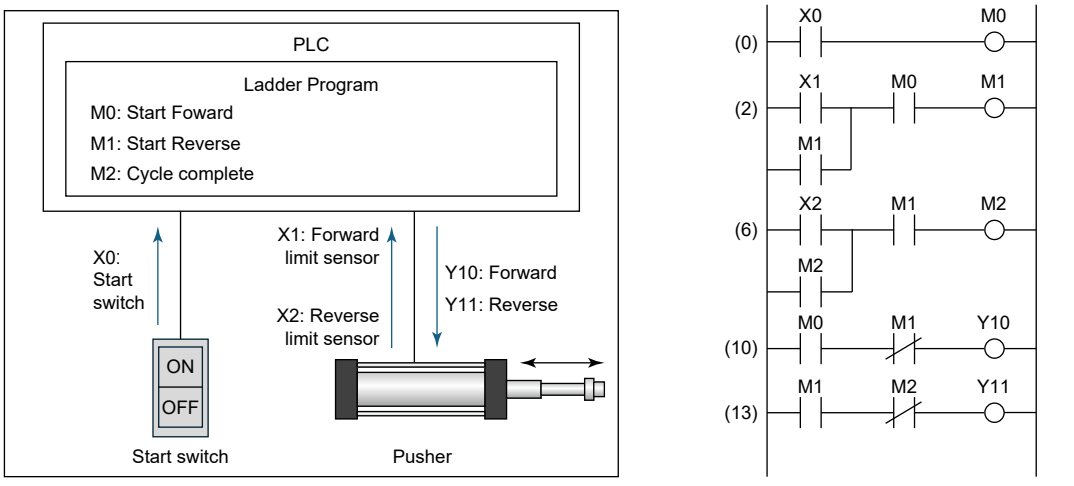
Abstract

Ladder logic is a programming language for describing control operations executed by a Programmable Logic Controller (PLC) and is used to control various equipment. When a control systems engineer modifies equipment, they need to analyze the control logic written in ladder logic (hereafter, "ladder program") and understand the processing details, but this task takes a lot of work. To address this, Mitsubishi Electric developed a technology that uses Large Language Models (LLMs) to analyze ladder programs and generate processing description. It features analyzing not only the ladder program but also PLC log data, and explaining the ladder program's processing in the order of signal changes. As a result, the sequence in which devices within the equipment are controlled is explained clearly, reducing the workload on engineers when it comes to decoding ladder programs.

1. Introduction

PLCs are used to control various equipment. When a PLC controls equipment, it repeatedly receives signals from devices within the equipment, processes the signals in the control logic, and outputs signals to devices within the equipment. There are several programming languages for writing control logic, but in Japan the programming language known as ladder logic is used particularly often.

Using Fig. 1, we outline equipment control using a ladder program. Figure 1(a) shows equipment composed of a PLC and devices (start switch, pusher), and X0, Y10, etc. in the figure are input/output signals exchanged between the PLC and the devices. In equipment comprising a PLC and devices like this, the PLC receives input signals from devices, processes those signals in the ladder program, and passes output signals to devices to control them.



(a) Equipment (b) Ladder program to control equipment

Fig. 1 Example of an equipment configuration and its ladder program

Figure 1(b) is an example of a ladder program that controls the equipment in Fig. 1(a). In a ladder program, horizontal lines are drawn between the left and right vertical lines; the condition is written on the left side of a horizontal line, and the processing on the right side, indicating what processing is performed under which conditions. For example, in item (0) of Fig. 1(b), the left portion indicates the condition "when X0 is 1," and the right portion indicates the process "set M0 to 1." Similarly, in item (10) of Fig. 1(b), the left

portion indicates the condition “when M0 is 1 and M1 is 0,” and the right portion indicates the process “set Y10 to 1.” Combining the conditions and processes in items (0) and (10) of Fig. 1(b) yields control: “set Y10 to 1 when X0 is 1 and M1 is 0.” With this control, when the equipment’s start switch is turned ON, the pusher moves forward until it reaches the forward end.

In this way, ladder programs are difficult to analyze because the values of multiple input signals and the order in which they change affect the processing, and the processing related to a single output signal is written across multiple parts of the ladder program.

2. Challenges in Understanding Ladder Program Control

When a control systems engineer modifies equipment or estimates the cause of an equipment malfunction, they need to understand how the equipment is controlled by the ladder program. However, for equipment that has been in use for a long time, the control systems engineer who originally designed it may no longer be available due to retirement, etc., making it necessary to analyze the ladder program.

Figure 2 is an example represented in text the result of a human decoding the ladder program in Fig. 1(b). In this text, you can see in what order the equipment’s signals change and which parts of the ladder program cause those signal changes. The steps in the text correspond to the numbers written on the left of Fig. 1(b). With such an explanation of the processing, a control systems engineer can determine which parts of the ladder program to check and modify when renovating equipment or estimating the cause of malfunctions.

When the user turns ON the start switch, X0 changes to 1, causing the process at step 0 of the program to set M0 (Start Forward) to 1. When M0 changes to 1, the process at step 10 sets Y10 (Forward) to 1, and the pusher begins to move forward.

When the pusher reaches the forward end, X1 (Forward End Sensor) changes to 1, and the process at step 2 sets M1 (Start Backward) to 1. When M1 changes to 1, the process at step 13 sets Y11 (Backward) to 1, and the pusher begins to move backward. Additionally, the process at step 10 changes Y10 (Forward) to 0, stopping the pusher's forward motion.

When the pusher reaches the backward end, X2 (Backward End Sensor) changes to 1, causing the process at step 6 to set M2 (Cycle Complete) to 1. When M2 changes to 1, the process at step 13 changes Y11 (Backward) to 0, and the pusher's backward motion stops.

When the user turns OFF the start switch, X0 changes to 0, causing the process at step 0 to reset M0 (Start Forward) to 0. When M0 changes to 0, the process at step 2 resets M1 (Start Backward) to 0. Finally, when M1 changes to 0, the process at step 6 resets M2 (Cycle Complete) to 0.

Fig. 2 Example of a manual analysis of a ladder program

However, as stated in Chapter 1, decoding ladder programs is difficult to analyze a large ladder program and generate an explanation like the one shown in Fig. 2.

Therefore, we developed a technology that uses LLMs to generate a processing description for ladder program. Generated by this technology, the processing description for ladder program clearly shows the order in which devices within the equipment are controlled, which is a key feature.

3. Method for Generating the Processing Description for Ladder Program

When generating a processing description using an LLM, the explanation is basically generated in order from the top of the ladder program. Therefore, if only the ladder program is used as input data to an LLM to generate a processing description, it cannot generate an explanation like that shown in Fig. 2, where the order of processing carried out by the ladder program is clear. This technology addresses this issue by using not only the ladder program but also PLC log data as input. Table 1 shows the steps for generating the processing description for ladder program and the input and output data at each step.

Table 1 Procedure for generating a processing description using this technology

No.	Procedure	Input data	Output data	Explanation
1	Splitting the ladder program	Ladder program	The split ladder program	Ref. 3.1
2	Extraction of signal change order	Log data	Signal change information at each time	Ref. 3.2
3	Mapping signal changes to the ladder program	Output data No. 1 and No. 2	Relationship between signal changes and the ladder program	Ref. 3.3
4	Generation of processing description	Output data No. 3 and meaning of each signal	Program description for ladder program	Ref. 3.4

3.1 Splitting the ladder program

The analysis result in Fig. 2 first explains the processing at step 0 of the ladder program, and then explains that the processing at step 10 of the ladder program is performed next. In a ladder program, processing is performed only where the condition is satisfied; therefore, as in this example, the order in which processing appears in the program may not match the order in which processing is actually executed.

Accordingly, this technique splits the ladder program as shown in Fig. 3 to make reordering possible. Splitting is performed at the ladder block level, which is a combination of the processing content and the condition under which that processing is performed. Then, to allow reordering in the correct sequence at a later stage, the condition part and the processing part are also split.

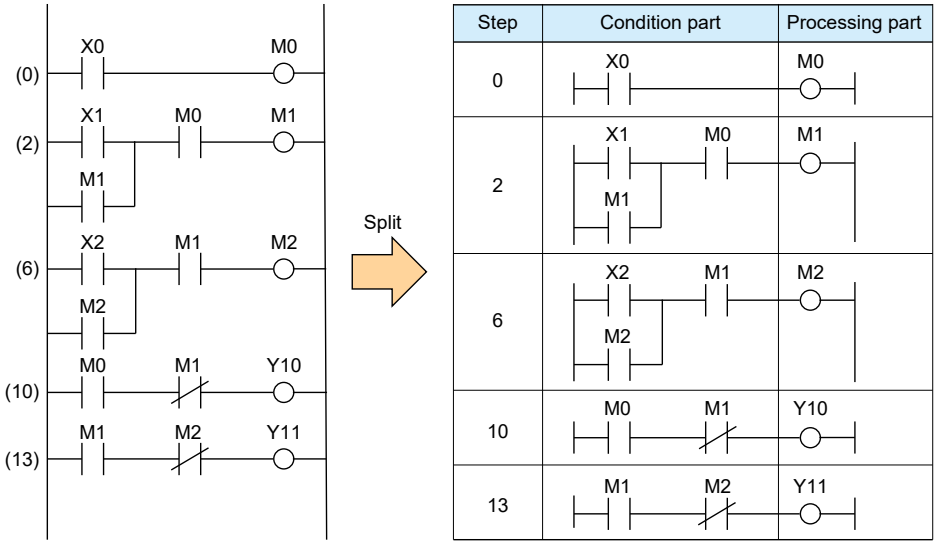


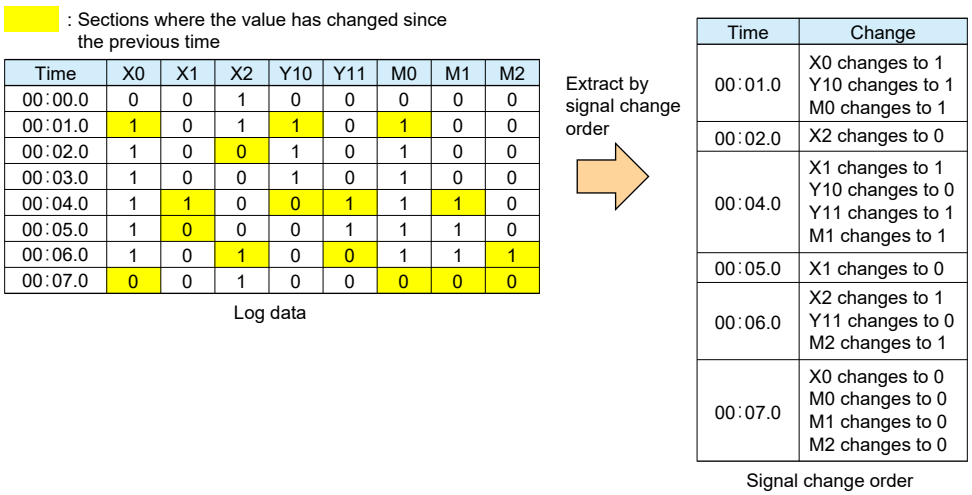
Fig. 3 Ladder program partitioning

3.2 Extraction of signal change order

The order of input signal changes is shown in the decoding results in Fig. 2—for example, X0 becomes 1 first, and then X1 becomes 1. Such changes in input signals occur due to device motion, and the ladder program alone does not indicate when or how input signals change.

Therefore, this technique uses PLC log data as input in addition to the ladder program. PLC log data records the state of each signal along with timestamps during equipment operation, and can be represented in tabular form as shown in Fig. 4. For example, in Fig. 4, the row where the time column is 00:01.0 shows a value of 1 in the X0 column, from which we can read that X0 is 1 at time 00:01.0.

In this technique, by extracting from the log data the parts where each signal's value has changed from the previous timestamp (as in Fig. 4), we create information about the signal change order—indicating when and how signals changed—which is required to generate the processing description.



Time	X0	X1	X2	Y10	Y11	M0	M1	M2
00:00.0	0	0	1	0	0	0	0	0
00:01.0	1	0	1	1	0	1	0	0
00:02.0	1	0	0	1	0	1	0	0
00:03.0	1	0	0	1	0	1	0	0
00:04.0	1	1	0	0	1	1	1	0
00:05.0	1	0	0	0	1	1	1	0
00:06.0	1	0	1	0	0	1	1	1
00:07.0	0	0	1	0	0	0	0	0

Time	Change
00:01.0	X0 changes to 1 Y10 changes to 1 M0 changes to 1
00:02.0	X2 changes to 0
00:04.0	X1 changes to 1 Y10 changes to 0 Y11 changes to 1 M1 changes to 1
00:05.0	X1 changes to 0
00:06.0	X2 changes to 1 Y11 changes to 0 M2 changes to 1
00:07.0	X0 changes to 0 M0 changes to 0 M1 changes to 0 M2 changes to 0

Fig. 4 Signal change extraction process from log data

3.3 Mapping signal changes to the ladder program

In the decoding results in Fig. 2, the signal changes are explained by associating them with the corresponding parts of the ladder program. To generate this kind of explanation, the results of splitting the ladder program in Section 3.1 needs to be associated with the results of extracting signal changes in Section 3.2.

In this technique, we extract candidate associations and confirm whether those candidate associations are correct by verification using an LLM.

In the procedure for finding candidate associations, based on the signal changes extracted in section 3.2, we extract from the ladder program split in section 3.1 those programs in which both the signals in the condition part and the signals in the processing part change at the same timestamp. In the example in Fig. 5, at time 00:01.0 three signal changes occur; the ladder programs that include these changes in both the condition part and the processing part are the ladder programs in step 0 and step 10. In these two programs, one can predict that the processing was performed because the value of the signal serving as the condition changed, potentially causing other signal values to change.

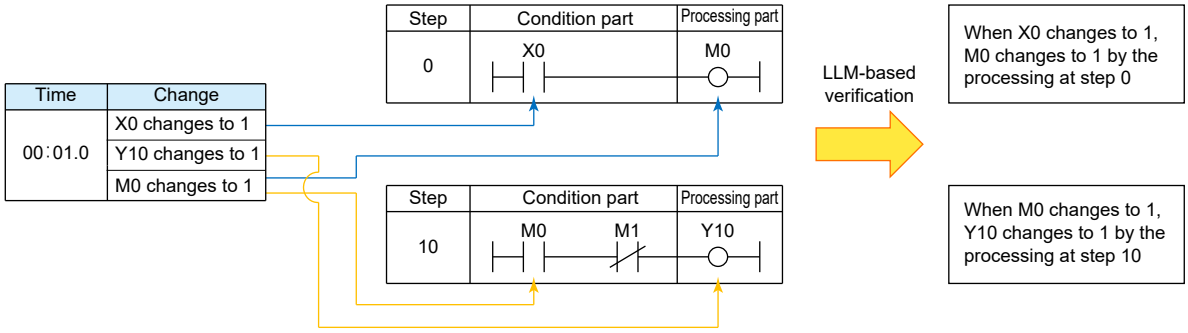


Fig. 5 Associating signal changes and corresponding ladder block

In the procedure for verifying candidate associations using an LLM, we provide the signal values and the ladder program to the LLM, and check whether the signal values change due to processing by the ladder program. The reason this procedure is necessary is that the candidate associations extracted above are not necessarily correct. For example, in the program of step 10 in Fig. 5, the condition part includes the signal M1, and depending on the value of M1, even if M0 becomes 1, Y10 does not become 1. Furthermore, if multiple candidate processing steps exist in the ladder program that could change the same signal value, this method is needed to determine which candidate actually changed the signal.

3.4 Generation of processing description

Finally, we combine the explanations obtained in section 3.3 to generate the processing description for ladder program. Here, by adding not only the signals but also the meaning of each signal—for example, “X0 (start switch)”—to the explanation, we make it easy to understand the relationship between the program and equipment operations.

4. Validation

We conducted a validation to confirm that this technique is able to generate explanations that correctly describe the processing content of the ladder program, and clear explanations that describe the processing in the order of signal changes. In the validation, we used gpt-4o (2024-05-13) as the LLM.

4.1 Validating that explanations correctly describing the processing contents can be generated

Using this technique, we generated the processing description for ladder program for the ladder program in Fig. 1(b) and compared the result with the human decoding in Fig. 2, to verify whether the processing description was correctly described in the generated explanations. Figure 6 shows the result of generating the processing description for ladder program using this technology. When comparing Fig. 2 and Fig. 6, causes of signal changes such as “pusher reaches the forward end” are not described in Fig. 6 because the input data lacks that information; however, from the meaning of each signal—such as “Start Forward” and “Forward End Sensor”—an approximate operation can be inferred. Otherwise, Fig. 2 and Fig. 6 present equivalent content, and the processing content is described correctly in the generated explanations.

```
When X0(Start Switch) changes to 1, the process at step 0 set M0(Start Forward) to 1.
When M0(Start Forward) changes to 1, the process at step 10 set Y10(Forward) to 1.

After 3.0 seconds, when X1(Forward End Sensor) changes to 1, the process at step 2 set M1(Start Backward) to 1.
When M1(Start Backward) changes to 1, the process at step 13 set Y11(Backward) to 1.
Additionally, the process at step 10 set Y10(Forward) to 0.

After 2.0 seconds, when X2(Backward End Sensor) changes to 1, the process at step 6 set M2(Cycle Complete) to 1.
When M2(Cycle Complete) changes to 1, the process at step 13 set Y11(Backward) to 0.

After 1.0 seconds, when X0(Start Switch) changes to 0, the process at step 0 set M0(Start Forward) to 0.
When M0(Start Forward) changes to 0, the process at step 2 set M1(Start Backward) to 0.
When M1(Start Backward) changes to 0, the process at step 6 set M2(Cycle Complete) to 0.
```

Fig. 6 Generation results of program descriptions using this technology

4.2 Explanations of processing content generated in the order of signal changes —validation

We compared the processing description generated using this technique, with the processing description generated using only the ladder program as input data, and confirmed the effectiveness of the procedure described in Chapter 3. Figure 7 shows the processing description generated using only the ladder program as input data.

```
In step 0, if X0 is 1, M0 is set to 1.
In step 2, if X1 is 1 or M1 is 1, and M0 is 1, M1 is set to 1.
In step 6, if X2 is 1 or M2 is 1, and M1 is 1, M2 is set to 1.
In step 10, if M0 is 1 and M1 is 0, Y10 is set to 1.
In step 13, if M1 is 1 and M2 is 0, Y11 is set to 1.
```

Fig. 7 Generation results of program descriptions using only ladder program as input

Given that the explanation shown in Fig. 7 is created in order from the top of the ladder program, it is not possible to determine which parts of the program perform processing in what order. Moreover, it is also unclear in what order the changes in each signal occur. On the other hand, as shown in Fig. 6, the description of the processing content generated using this technology resolves these issues and describes the ladder program’s processing content in the order of signal changes in the log data. With such a description, it is possible to reduce the workload of engineers in analyzing the ladder program.

5. Conclusion

We described a technology that leverages LLMs and, by mapping signal changes extracted from log data to the ladder program, generates clear explanations of the control sequence of devices within the equipment. By utilizing this technology, the workload of engineers in analyzing ladder programs can be reduced.

Going forward, we will consider further developed technologies based on this approach, such as a chatbot that answers questions about the processing content and functions that explain differences from the specifications.