A Low-Code Development Method for Manufacturing Facilities

Authors: *Takaaki Abe*, Satoshi Noguchi**

**Information Technology R&D Center*

Abstract

The distinguishing feature of the low-code development method for manufacturing facility control programs is that control programs for realizing cycle operation are generated from flowcharts widely used as a design technique for cycle operation. Since the generation of control programs can be done without changing the conventional design technique, the method enables reduction of implementation man-hours needed for the control program, without increasing design man-hours. In the future, we will move forward with development aimed at commercialization by applying this method prototype to development of in-house facilities, and confirming its effectiveness.

1. Introduction

In recent years, demand has been steadily rising for factory automation due to social problems such as the shrinking working population. However, development of facilities for factory automation requires many man-hours, and thus there is a need for greater efficiency in facility development. Among facility development tasks, the task of implementing a control program for controlling facilities is in particular need of greater efficiency. This control program implementation work includes many simple tasks where implementation is done by repeatedly writing similar types of processing based on facility design. Therefore, there is a need for technology to generate control programs from facility designs in order to make this kind of work more efficient. This paper examines a low-code development method enabling generation of control programs from facility designs.

2. Low-Code Development Method for Control Programs

2.1 Control programs to be generated and their design technique

Manufacturing facilities perform automatic production based on cycle operation in which a certain series of actions is repeatedly executed. Therefore, control programs must implement cycle operation by coding the operation sequences and operation conditions of each machine provided in the facility. This cycle operation program is a crucial program that directly affects the facility's production capacity because it determines the movements of the facility in automatic production. Also, this program is implemented by repeatedly writing similar processing, so there is a need for greater efficiency in implementation. The correspondence relationship between design content and implementation content for cycle operation programs has previously never been sorted out, and no efficient techniques have been provided for generating programs from designs. Thus, this paper examines a low-code development method for generating programs for this cycle operation. This section describes an implementation technique that is widely used in cycle operation programs, and the associated design technique.

Control programs for facilities in Japan are frequently implemented using ladder language, one of the languages specified by the IEC 61131-3 standard for PLCs. Figure 1 shows an example ("Control program" section on the right half of the figure). In ladder language, a control program is implemented by describing relay circuit type processing called circuit blocks, and combining multiple circuit blocks. In ladder language, a notation method called state transition format is well known as an implementation technique for cycle operation[(1)]. This notation is composed of two parts, a state transition section and an output section. The state transition section describes circuit blocks that sequentially advance the states of the facility's cycle operation based on values such as sensor signals. More specifically, as shown in the state transition section in Fig. 1, it describes a circuit block which sequentially turns ON state variables (M0 to M2) each time a condition such as a sensor signal (X1) (yellow box in Fig. 1) going ON is satisfied, and then turns OFF all state variables when the final state variable goes ON. The result of this is a program for cycle operation

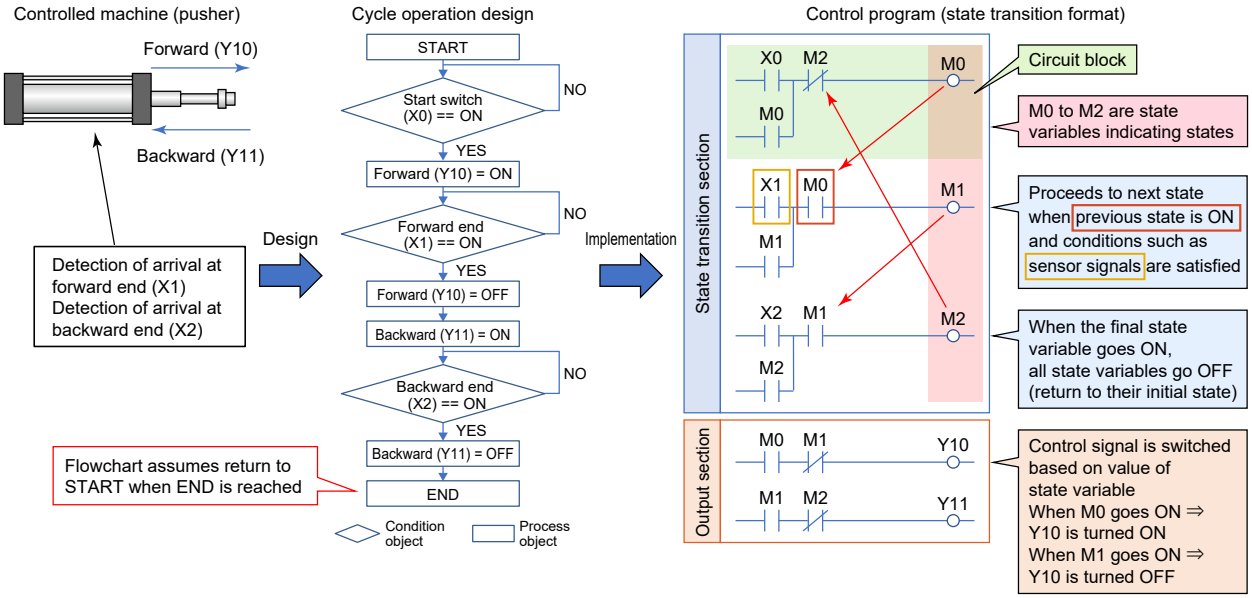

Fig. 1 An example of a cycle operation design and program

which returns to the initial state after the state transitions, and repeats state transitions from the initial state.

The output section describes circuit blocks that turn ON/OFF control variables corresponding to each machine based on the value of state variables in the state transition section. As shown in Fig. 1, the described circuit block is such that when the state variable M0, which is the ON condition for the control variable Y10, goes ON, Y10 is turned ON, and when the state variable M1, which is the OFF condition for Y10, goes ON, Y10 is turned OFF. Cycle operation of each facility is implemented using state transition format consisting of these state transition sections and output sections.

In many cases, specifications for this cycle operation are designed using a flowchart like that in Fig. 1 (section labeled “Cycle operation design”). This flowchart assumes that the flow of the sequence is repeated by executing sequential processing from START, and then returning to START when END is reached. The flowchart shown in Fig. 1 is an example representing cycle operation of a pusher operated by a pneumatic cylinder.

2.2 Issues with the low-code development method

There are two issues for enabling low-code development of control programs. First, it must be possible to design cycle operation with a flowchart, which is the conventional design technique shown in Fig. 1. If the low-code development method requires a design technique significantly different from that used before, there is a possibility that design man-hours will increase, and efficiency will not be improved. Also, if the design technique is close to the conventional approach, it will be possible to seamlessly shift to development using this method.

Second, it must also be possible to generate control programs in a form close to previously implemented control programs. Source code for control programs is read and expanded by people other than the implementer, such as maintenance staff. Therefore, if the forms of the generated control programs change, it will hinder this work and may interfere with maintenance, etc.

Due to the above considerations, we have devised a method for generating control programs in state transition format from flowcharts, which are the conventional design technique, in order to resolve these two issues.

2.3 Control program generation step

With the low-code development method, the flowchart design content and circuit blocks in state transition format are redefined as combinations of patterns, and a control program in state transition format is generated from a flowchart by mapping the flowchart patterns to the control program patterns based on a correspondence relationship. This section describes that correspondence relationship, and then describes the technique for generating control programs based on that correspondence relationship. To simplify the

explanation in this section, the patterns in flowchart design content and the patterns in generated circuit block are limited to those shown in Fig. 1. There will be multiple patterns when applying the method to development of actual facilities. Examples of patterns include: cases where the direction of flow branches at a condition object, and cases where a timer is used as a condition.

First, circuit blocks for state transition sections are generated from condition objects in the flowchart. As noted in section 2.1, the state transition section has the characteristic that the system transitions to the next state when the condition is satisfied. Therefore, a condition object indicating flow to the next object when the condition is satisfied is placed into correspondence with a state transition section in state transition format. Next, a circuit block for the output section is generated from the process object of the flowchart. The output section has the characteristic that it switches a control variable ON/OFF based on the value of state variables in the state transition section. Therefore, the output section in state transition format is placed into correspondence with the process object indicating rewriting of the value of a control variable. The switching of the control variable is done in multiple process objects, so multiple process objects and a single circuit block of the output section are placed into correspondence. With the technique examined here, circuit blocks are generated in the three steps indicated below based on these correspondence relationships (Fig. 2).

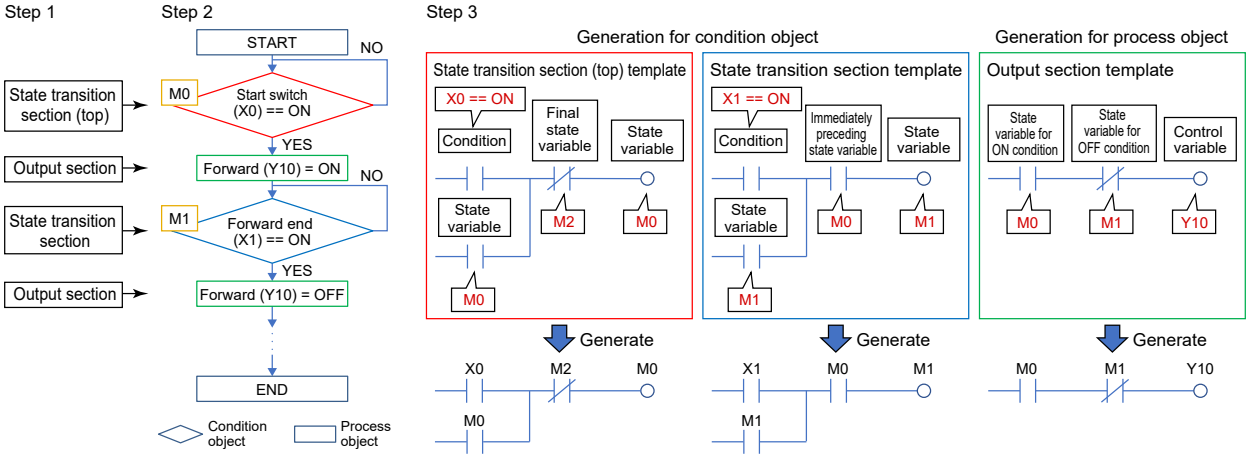

Fig. 2 Control program generation process

(1) Step 1: Flowchart analysis

The flowchart is analyzed, and for each object in the flowchart the circuit block pattern to be generated is determined based on the type of object and the line connection information between objects. As shown in Fig 2, it is determined, based on the type of object, that the pattern to be generated is a circuit block of a state transition section for a condition object, and a circuit block of an output section for a process object. The circuit block at the top of the state transition section includes processing to reset all states, and is different from circuit blocks not at the top. Therefore, of the condition objects, the one at the top of the flowchart is determined from line connection information, and it is determined to be the pattern for “state transition section (top)” as indicated on the left side of Fig. 2.

(2) Step 2: Variable assignment

A single state variable is needed for a single process of a state transition section, and thus a single state variable is assigned to each condition object. State variables are assigned in sequence, starting from M0, to the condition objects in Fig. 2.

(3) Step 3: Generation of circuit blocks corresponding to each object

Based on the flowchart information and allocated state variable information, circuit blocks are generated that reflect the contents of the flowchart from the circuit block templates corresponding to the patterns identified in Step 1. Circuit blocks for condition objects are generated by identifying the variables to be set in the template, based on the conditions indicated in the condition objects, information on line connection between objects, and other factors, and then setting those variables. Circuit blocks for process objects are generated by identifying the necessary variables based on line connection information relating to the multiple process objects for each control variable, and then setting those variables.

In this way, a control program in state transition format is generated by generating circuit blocks corresponding to flowchart design content, and arranging the generated circuit blocks to correspond with the flowchart.

3. Conclusion

This paper has presented a low-code development method for generating control programs for cycle operation from flowcharts. Since generation can be done using this method without changing the conventional design technique, the method enables reduction of implementation man-hours needed for the control program, without increasing design man-hours. Going forward, we will verify the effectiveness of this method, and strive for commercialization.

References

(1) Kumagai, H.: Handbook: Standard Sequence Programs using PLC, Techniques for Creating Ladder Diagrams to Control Equipment, Nikkan Kogyo Shimbun (2021)