

Model-Based Design of PLC Programs

Mitchell M. Tseng (1), Matthias Schreyer
Department of Industrial Engineering and Engineering Management,
The Hong Kong University of Science & Technology
Kowloon, Hong Kong

Abstract

Programmable logic controllers (PLC) have been one of the important players in manufacturing systems and many other applications with sales of 15 million PLCs worldwide every year. However, our research has revealed that the programming of PLC based control systems is still very much relying on trial-and-error. In addition, current PLC based control systems are not properly designed to support the growing demand for flexibility and reconfigurability of manufacturing systems. This paper will report a systematic approach to model and design PLC based control systems for reducing the uncertainties in managing the control software development, increasing the flexibility of the systems, and facilitating the use of simulation technology for systems validation. This approach involves systematic design principles, formal modeling techniques, and a generic software component library structure. An example of using the proposed approach is given to illustrate the design of PLC based control systems for a material handling systems.

Keywords: Design, Modeling, Programmable logic control

1 INTRODUCTION

Programmable Logic Controller (PLC) technology, invented more than 25 years ago, has been widely used in industry including manufacturing systems, transportation systems, chemical process facilities, and many others. Although PC based (soft-logic) control [1] has started to come into place, PLC based control will remain the technique, to which the majority of industrial application will adhere, due to its high performance, low price and increased reliability in harsh environments. Moreover, according to a study on the PLC market of Frost and Sullivan [2], an increase of the annual sales volume to 15 million PLCs per year with the hardware value of more than 8 billion US dollars has been predicted, though the prices of computing hardware is steadily dropping.

2 KEY ISSUES IN PLC BASED CONTROL DESIGN

There are several key issues in the design of PLC based control systems that need to be addressed.

The design of PLC based control systems is characterized by uncertainty and long development time. Uncertainty emerges from the enormous amount of specific requirements, such as software quality, hazard analysis requirements, and the hardware requirements imposed from the mechanical system to be controlled. In order to cope with uncertainty, more than 50% of the manpower allocated for the control system design is scheduled for testing and debugging. The long design lead-time is a result of a rigor sequential design process in industrial practice, in case system developers do apply a system methodology at all. In many cases the design of the mechanical manufacturing or transport system and the control system are passed on to different subcontractors with less or no communication.

A major problem, impelling the need for a systematic design methodology, is the increasing software complexity in large-scale projects alongside with the permanently growing fraction of software life-cycle cost. A common PLC application has often to handle complex systems with more than 10,000 control nodes. In industrial automation, 80-90% of the costs are going into

software maintenance, debugging, adaptation and expansion to meet changing customer requirements [3].

The increased demand for reconfigurability of manufacturing systems in quick response-markets imposes an additional requirement on control systems. The raising pace of changes in the manufacturing systems requires frequent adaptations of control programs. A major challenge is therefore to provide enabling technologies that can economically reconfigure manufacturing control systems in response to changing needs and new opportunities [4].

Consequently, the need to strive for higher software quality, to cope with increased complexity, and to enable the quick and economical reconfiguration of manufacturing control systems require advanced concurrent design concepts, modern control technologies, and new computer-aided modeling and design tools. This paper presents a design approach to design PLC based control systems in order to reduce the uncertainties in managing the control software development, increasing the flexibility of the systems, and facilitating the use of simulation technology for systems validation. The design approach includes formal top-down modeling techniques that are well known in real-time control research, but less known in industrial practice. An advanced design methodology also implies integrated modeling of the manufacturing system and the control system. Modeling of the manufacturing system facilitates off-line testing and simulation of the control system and should not cause additional cost though it may further consume design time.

A reusable component library will help to ease the design and configuration of domain-specific systems. In this context, domain engineering targets at the design for reusability of software systems [5].

3 DESIGN APPROACH FOR INTEGRATED MANUFACTURING AND CONTROL SYSTEM

3.1 Current practice in control software engineering

In most cases the software implementation of the control systems occurs during the installation of the manufacturing system. PLC programming is left over to control engineers or technicians who may have less knowledge about methodical and systematic design and programming. Only in large-scale and demanding projects,

dedicated service provider for information and automation systems are requested. These specialists follow a rigorous sequential project methodology. Coarsely, the project steps are functional specification, hardware design, manufacturing and procurement, control system development and integration, factory acceptance test (FAT), and delivery including documentation and installation. The allocated manpower is based on empirical data from previous projects [6]. However, the delegation of the control system design to dedicated subcontractor embodies the problem of less interaction and communication between the design team of the mechanical system and the control system.

3.2 Integrated design approach

In order to provide a coherent information flow and to achieve a reduction in the design lead-time we propose an integrated design approach based on axiomatic design theory [7].

Axiomatic design is a simple, but powerful design methodology. It supports the design process with a general set of design rules forcing the designer to think systematically from a top-down perspective. Thereby, the two axioms, (1) Independence Axiom and (2) Information Axiom, guide the designer to generate a “good” solution and to select the best one among alternatives. Conceptually, the axiomatic design approach divides the design process into four separated semantic and syntactical perspectives, the domains of customer

requirements (CR), functional requirements (FR), design parameters (DP), and process variables (PV). Although originally developed for mechanical systems engineering, axiomatic design has been applied to many other design contexts, including software engineering. Accordingly, the domains may have different interpretations in diverse design contexts.

In the design of manufacturing system, the DPs describe the processes in a solution-neutral format, whereas the PVs indicate the selected resources, including software systems, hardware systems and human resources. The DPs in the manufacturing system design therefore impact the FRs of the control system to be designed. These are processes, control actions, and temporal requirements. Enabler (DPs) of the processes and control actions are software objects and states describing the abstract structure and system’s behavior. This abstract representation will finally be mapped into the implementation domain of the control system (PVs) indicating the program constructs and program code of the particular language, and the run-time platform.

Additionally, common user requirements, such as best performance, reliability and reconfigurability, are superimposing the FRs from manufacturing processes to be implemented by the control system. Thus, the FR domain of the control software design process is pooling the requirements, derived from the manufacturing system configuration, and the desired control system attributes.

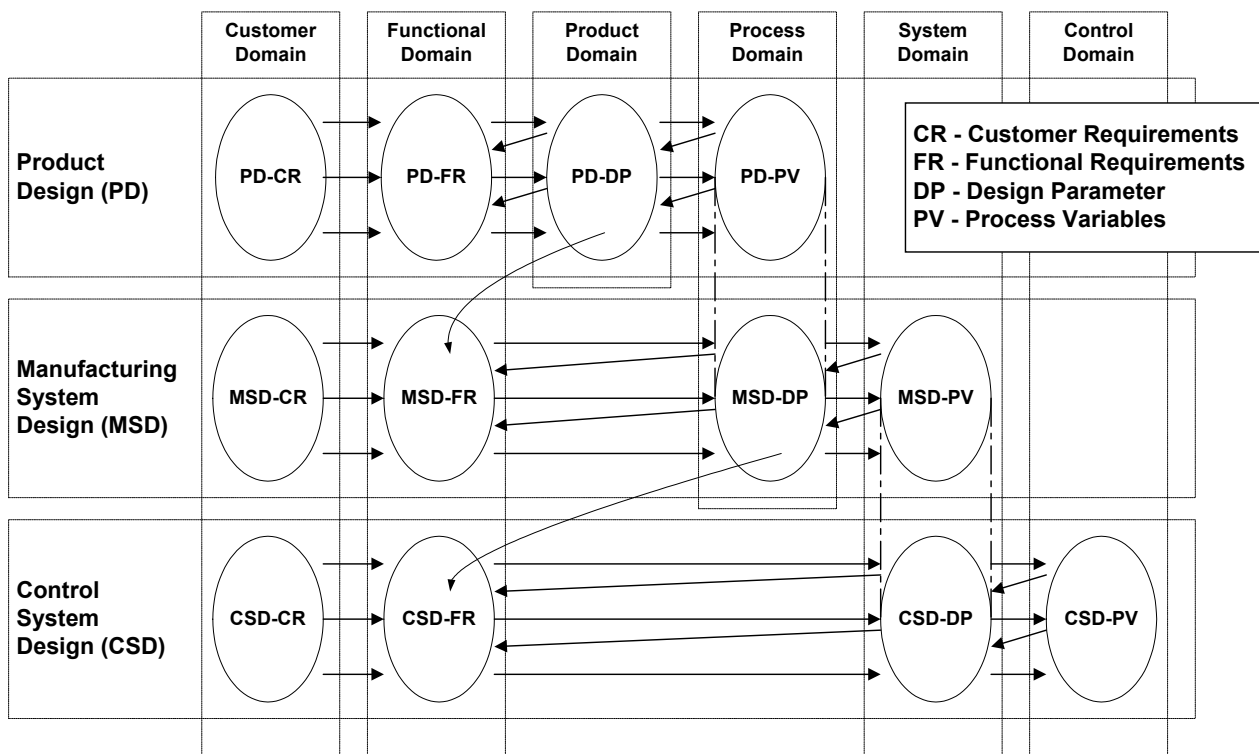


Figure 1: Concurrent design approach based on axiomatic design.

3.3 Formal Modeling

Capturing all kind of design decisions during the development process requires various notations, each of them indicating a certain point of view. Commonly, three views are selected to describe the functionality, behavior and the structure of a software design. This F-B-S consideration gives answer to the three basic questions:

- **F:** *What to do ?*
- **B:** *When, or in response to what, to do ?*
- **S:** *Where to do ?*

The functionality indicates the *what-to-do* in the system to be designed, and might be represented with functional-oriented notations, such as SADT [8], SA-RT [9], or activity charts [10] in a decomposed hierarchical tree structure. The activity hierarchy together with the flow of information constitutes the functional view.

The behavioral view specifies the dynamics of the system that is to say when and in response to what event the activities are triggered. The behavior could be represented with statecharts [10], various types of Petri nets [11], or formalized temporal logic [12].

The structural view provides a hierarchical decomposition into modules and the communication between them. Usually, those modules represent a natural mapping from physically existent objects, such as devices, network nodes and manufacturing subsystems.

In recent research on formal modeling, Selic et al. [13] began to combine real-time modeling and design with object-oriented formalism. Likewise, Douglass [14] augmented the Unified Modeling Language (UML) with real-time description techniques, for instance timing diagrams. Stewart et al. [15] developed a port-based model focusing on the structural design of real-time control systems.

4 PLC BASED CONTROL SOFTWARE DESIGN

4.1 Graphical Notations

Visual modeling notations are important to convey design ideas and design decisions. They also help to allocate the functional requirements and the design parameters.

In this research, we are employing statecharts to describe the behavior and module-charts, which are similar to those proposed in [7] to indicate the system structure (Figure 2). Statecharts as well as module-charts are nesting and orthogonal (independent state/module regions). Therefore, both notations are useful for a hierarchical decomposable design process. A module-chart is always accompanied with a statechart. In general, state transitions models are advantageous because they support directly a cognitive mapping from functional requirements to design parameter, here into states. In addition, state transition models may handle contingent behavior in a natural way. As a third notation, ladder logic charts are used to explicitly specify the Boolean conjunctions between incoming events. Though the ladder logic could be regarded as a redundant description to statecharts, we will reconsider it due to its high acceptance in industry.

4.2 Design Process

In the following the important design steps are summarized. Step 1 and 2 refer to the manufacturing system design process (Figure 2), whereas step 3 and 4

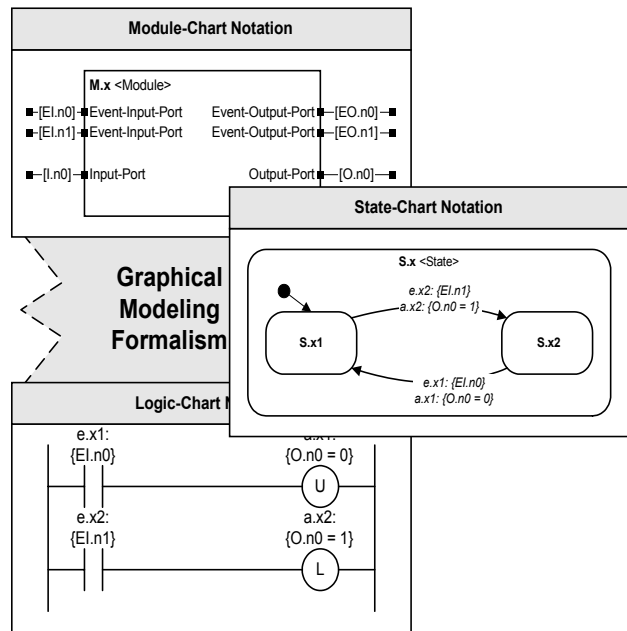


Figure 2: Notations for structure, behavior, and logic.

indicate the mapping procedure from the CSD-FR to the CSD-DP domain.

Step 1a. Create module architecture from MSD-FR and MSD-DP domains:

The modular architecture [7] of the manufacturing system indicates the physical components to be controlled. In case the modular architecture is not available a structural decomposition of the system exhibits the dynamic components.

Step 1b. Structural decomposition of the system to be controlled:

Structural decomposition of the identified physical components gains elementary functional modules or actors. I/O ports are used to identify the boundary of each module. They indicate the data flow between functional units and refer to the incoming event/sensor or outgoing event/actuator signals.

Step 2. Create an ontology to identify the task-neutral system's behavior (MSD-DP):

Define a suitable set of task-neutral observable states, in which the controlled manufacturing system could be. By this means, an ontology about the controlled system is created. The system states are a set of disjoint states of system components to be controlled, such as gripper, shuttle, rotary actuators, and linear drives. Each of those components has a discrete number of states (Figure 3). This step should be usually performed on the manufacturing system design stage since it describes the task-neutral behavior of the manufacturing system.

Step 3. Identify task-oriented CSD-FR and interpret them into transition of states (CSD-DP):

Identify the desired high-level functional requirements on highest level and interpret them into transition of states. For instance CSD-FR_x = "Transfer material bin from A to B" with CSD-DP_x = "Transfer state". In practice, each identified module has predefined high-level states, such as "on/off-state", "initializing state", "requesting state", "recovering state", *et cetera*. This step may be accompanied with selecting a statechart pattern of a domain database, wrapping up the task-oriented activities (Figure 4).

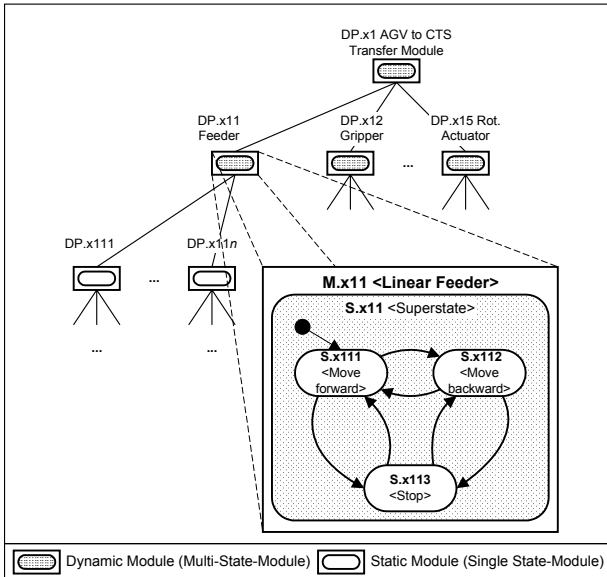


Figure 3: Deriving behavior from system architecture.

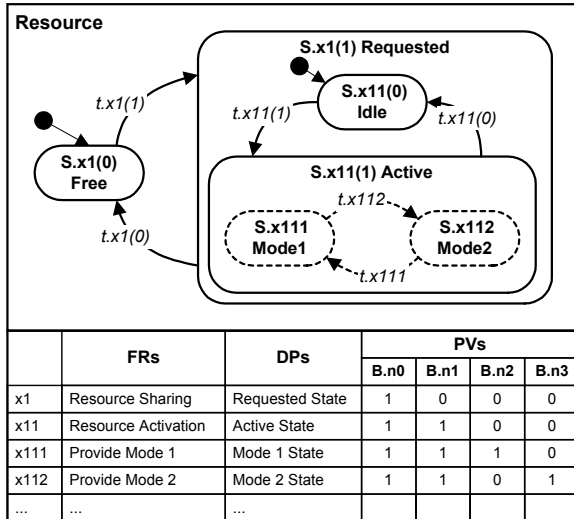


Figure 4: Statechart pattern for an abstract resource.

Step 4. Behavioral decomposition:

Statecharts support visually the behavioral decomposition. Hereby, the CSD-FRs describe the intended activities and commands that are going to be enclosed within states (CSD-DPs). Subordinate activities and states are identified through a zigzagging procedure between the CSD-FR and CSD-DP domains. A state is always accompanied by a triggering event and an internal or external action. An external control action creates a motion or state change of the manufacturing system. Therefore, it is causing a new physical state to be adopted by the machine or process. Internal actions are needed to facilitate the sequential or parallel execution of operations and the communication between modules of the control system. Activities have duration, whereas events and actions do not consume significant time. Activities are a composite of actions, which are irreducible.

Step 5. Decouple the design:

Modified design matrices, denoted as state transition tables, have been developed to model and analyze the interdependencies of modules and states (Table 2). The process of decoupling, segregates the design into a software structure with modules having less inter-modular communication, but high intra-modular interaction. In addition, explicitly modeling the interdependencies of modules and states raises the robustness of the design. The decoupling procedure assists to find a set of stable states among the usually numerous possible states of the control system. These stable states are desired operating states and cognizable errors states from which it is easier to recover in case of a faulty system behavior [16].

Based on the state transition notation we have developed a method to analyze the coupling between states and modules. The statechart description is mapped into a state transition table indicating the inputs, outputs and data elements of each state. The state transition table checks the interdependencies between modules and states.

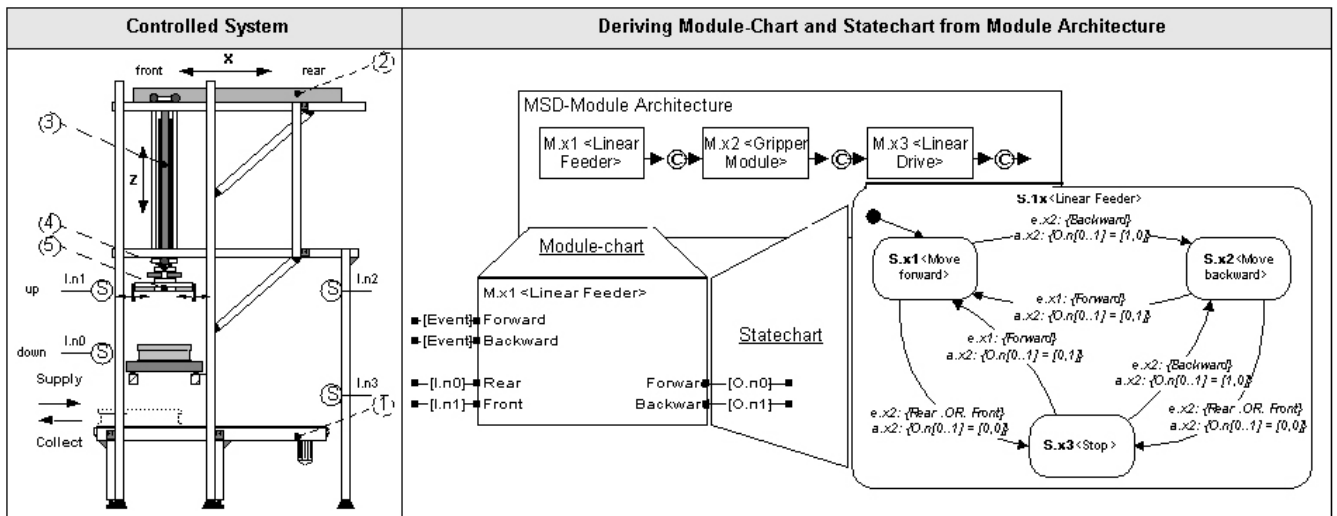


Figure 5: Robot for material supply and collection.

5 EXAMPLE

The developed design approach is illustrated by the means of a program segment of a transport system. The example shows that the entire semantic could be described in a concise and coherent way with better understandability and fewer elements compared to the original programming in relay ladder logic.

The functional specification and mechanical design of the material handling robot is given in Figure 5. The task-independent behavior of the robot is derived from the MSD-design process. The dynamic components are the feeder (1), horizontal drive (x-axis) (2), vertical drive (z-axis) (3), rotary actuator (4), and the gripper (5). These components are represented with a module-chart and statechart description to describe the I/O ports and the behavior. On the right-hand side of Figure 5 the feeding module illustrates the derived notations.

The material-handling robot might be controlled with a decentralized controller module. Therefore the next step implies the restructuring of the controller node and the modeled components to be controlled (Figure 6, left-hand side). Since the robot is a shared resource we may select a common resource state pattern from the database as depicted in Figure 4. The specific behavior is always wrapped up with a *requested* state in order to avoid accessing the resource at the same time. The task-oriented behavior is encapsulated within the *active* state. The controller module should provide two modes denoted with a dashed line since they are not task-neutral. In the *supply* mode material bins are transferred from the

external transport system (AGV) to the internal transport system. The *collect* mode accomplishes the reverse operation. Each of these modes consists of 11 elementary control actions in a determined sequence as shown on the right-hand side of Figure 6 for the first control actions of the supply mode.

After finalizing the design of the entire behavior the PLC code can be generated automatically. Table 2 visualizes a part of the computer-internal representation with the output pattern, the state interdependency, and input pattern in a state transition table.

6 CONCLUSION

This paper has presented an integrated model-based design approach for PLC based control. This approach aims at reduced development time and higher software quality. Testing and debugging of the PLC program is integrated in the manufacturing system design process. Visualized formal modeling techniques increase the understandability of the PLC programs, particularly, for non-experts with different perspective and expertise. Combined with the simple but powerful axiomatic design principles, this model-based design approach accelerates the generation of correct PLC program code.

7 ACKNOWLEDGEMENTS

The authors would like to acknowledge Rockwell Foundation (RAHK 97/98) for supporting this research.

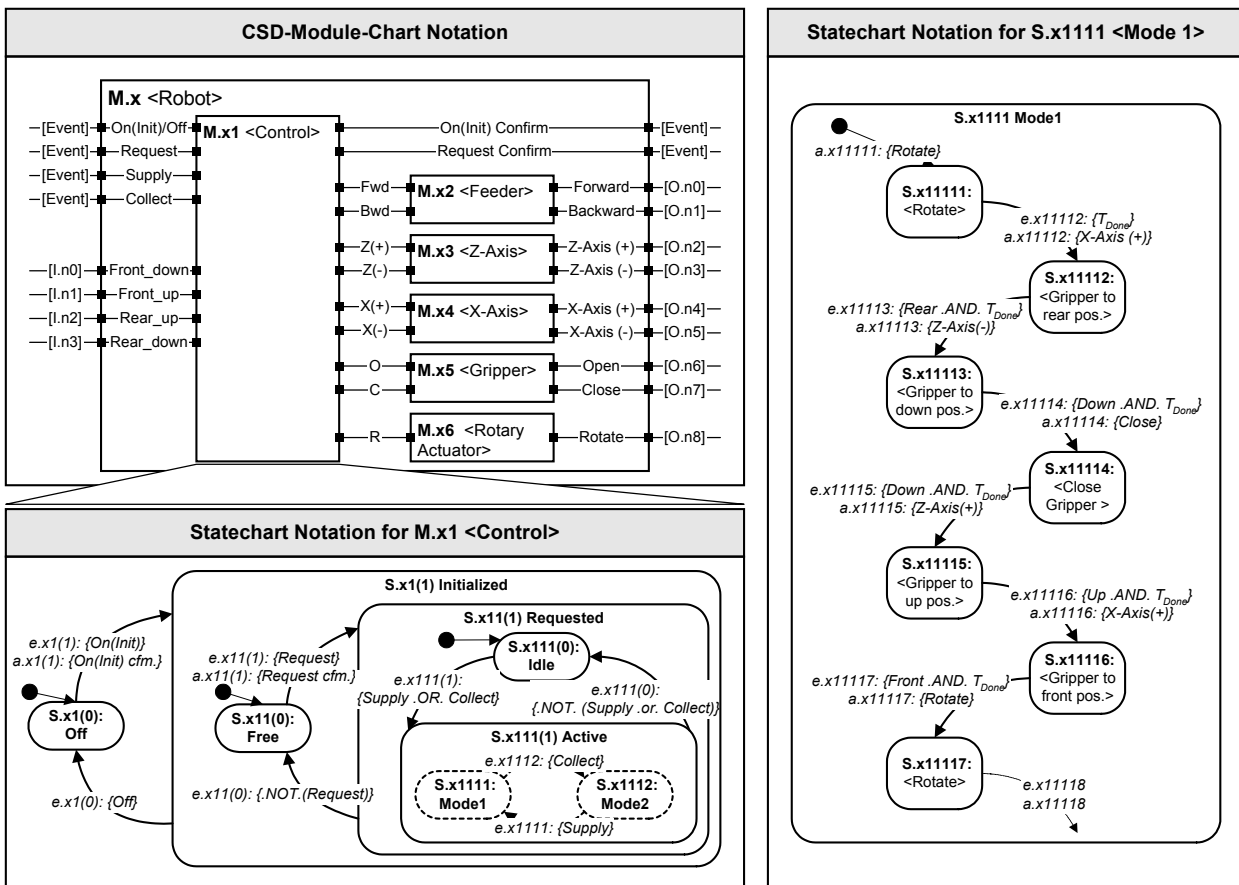


Figure 6: Part of the overall CSD-module-chart and -statechart notations.

