State models and PLC programming – Self-study materials

Author: Niko Siltala, niko.siltala@tuni.fi

Version	Date	Description
V1.2	2019-01-08	Update addresses
V1.1	2017-01-08	Update
V1.0	2016-01-18	Initial version

Contents

1 State models

1.1 Making a state model

UML State model. How state model is modelled and from what kinds of components it is composed of? Explanations for drawing marks. Guidance for State model components and making one can be looked from:
https://sparxsystems.com/resources/tutorials/uml2/state-diagram.html

1.1.1 Examples of state models:

http://www.conceptdraw.com/How-To-Guide/uml-state-machine-diagram

1.1.2 General notes for making state models for controls

State model should have an initial state (black ball), so that model is easier to start read (by others). This is important also for implementation. Make sure that the initial state is leading into some safe system state, or that if some action is occurring, it is leading the system towards safer state.

State model for controls usually runs in infinite loop, thus final state is not used. For the same reason, there should be at least one transition out from each state, so that application can always evolve. There should be at least one transition to every state, otherwise the state remains un-reachable.

I would progress according following steps when designing a state model for a control device:

1. To analyse what different (functional) states the system can be. I.e. what different functions or things happens in system. This will create the state space. States are always associated somehow to outputs and an operation or operational state of the machine. E.g. Stop, moving up, up, down, closed, …
    a. State should be uniquely named. Normally, in PLC program state must be uniquely identified, so this will come handy at later stages.
    b. Name should be descriptive, and provide some meaningful information to humans.
2. Next, analyse what conditions (=transitions) are connecting states. Transitions are causing the state changes. They are always triggered by some input (sensors, buttons (=operator action), …) or internal variable like expiration of timer (Timer.Q)
    a. One state can have several outgoing transitions, then trigger and [guard] define together into which branch is moved to.
    b. Inputs (sensors, selections, …) are usually associated to triggering conditions for transition!
    c. Guard conditions are written between brackets '[' … ']'. These are additional or external limiting conditions for the transition. Guard can block transition to happen even trigger becomes true.
    d. As rule of thumb, triggers are temporal, short time signals (e.g. push button, timer) and guards are normally some longer lasting inputs (e.g. mode switch, key switch, end sensor, timer)
    e. Triggers and guards are always evaluated as Boolean type value (true / false). Condition can also be some (Boolean logic) equation.
    f. Verify that there is at least one input and output transition associated into each state.

3. Setting/writing outputs happens **inside** states. E.g. we are in state X, so inside this state output *O_Up* = T(rue).
    a. Because of technical implementation, it is enough to mark only those (boolean) variables which are true in specific state. Value will be false in all the rest states.
    b. Outputs and actions are usually associated to states!
4. Other actions can be assigned to states. Starting and execution of timers, Set/reset for variables, setting value of variable, … Even conditional statements can be placed there, in case required.
5. State can have three functional code sections according the need. These are entry, do, and exit. The pseudo code section, tagged inside entry/ -label, is executed only one time once getting into the state. Do/ -section includes code that is executed on every PLC cycle. This is the default section, in case no label tag is set. Finally exit/ -section defines code that is executed once, when leaving the state. Pay attention the different functionality of these sections, e.g. in case of counters!

1.1.3 Drawing a state model

State models can be drawn on paper and then scanned or photographed. However, quite some SW modelling tools are available. Only a few are listed here. However, the main thing is that your model is clear and can be read without difficulty, and even more important, can be unambiguously interpreted.

Tool	url	Description
Umlet v14.3	http://www.umlet.com/	Simple drawing tool for making all kinds of UML diagrams.
Microsoft Visio		

Some generic ”drawing” tools can be utilized for making the models, such as Microsoft Powerpoint / Open office’s presentation tool.

1.2 State model examples

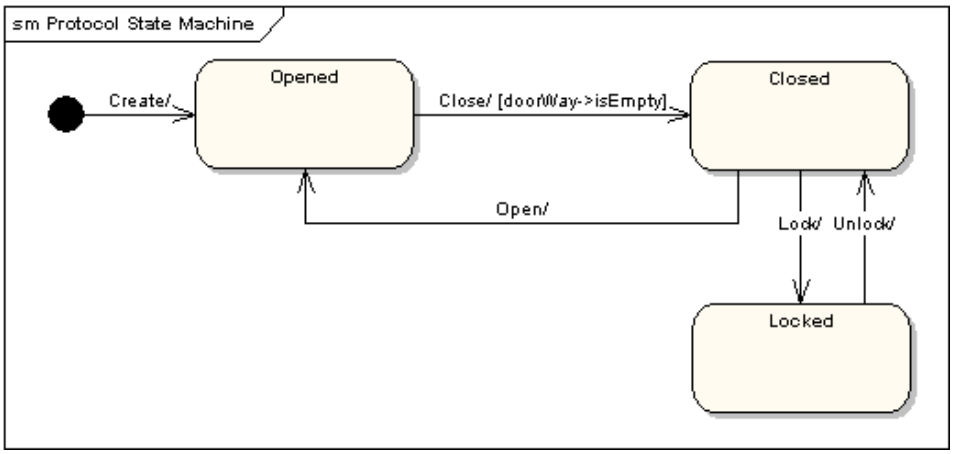

Figure 1 State model for a door [1]

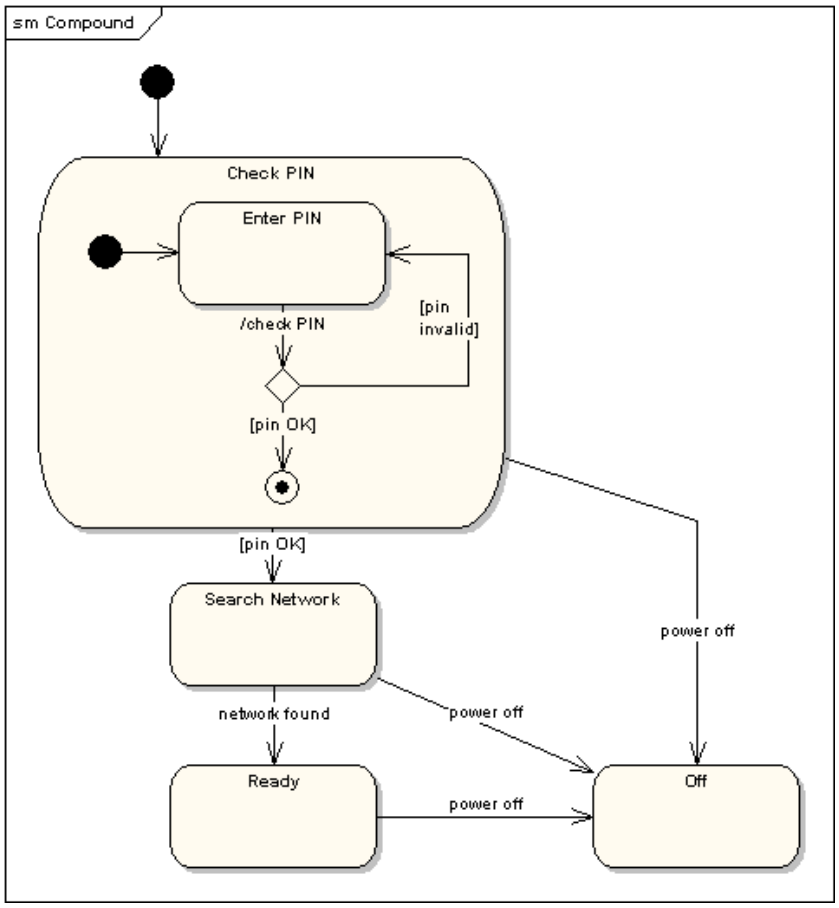

Figure 2 Entering PIN number and search for a network [1]

1.3 How the state model is transformed to PLC program code?

Next exercise we will study how state model can be transformed into different PLC code representations using different IEC 61131-3 languages. SFC is in practice direct graphical representation of state model. In ST utilises integer (or enumeration) representing the state and then Case and If..then – structures are used to make the model. Under following links there is represented very systematic way for LD.

eBook: Automating Manufacturing Systems; with PLCs. How state models can be applied in PLC programming and how state model turns into PLC code. Wide and lots of good material. LD (American) perspective.
http://engineeronadisk.com/book_plcs/plcsTOC.html

eBook: 12. STATE BASED DESIGN. Wide, good and extensive guide.
http://engineeronadisk.com/book_plcs/plc_sequ.html#53230

Solution: 12.1.2.3 - State-Transition Equations
http://engineeronadisk.com/book_plcs/plc_sequa2.html#16763

2 PLC programming

A set of programming languages for Programmable Logic Controller (PLC) [ohjelmoitava logiikka] are standardised in IEC-61311-3. The main PLC suppliers (generally the whole industry) are supporting widely these 4+1 languages. Suppliers might support all languages or only sub-set like only LD+ST.

What these languages then are? They are: Instruction list (IL), Structured Text(ST), Ladder Diagram(LD), Function Block Diagram (FBD). As fifth, there is Sequential Function Chart (SFC), which is aimed for defining sequences. The actual logical content of SFC is described with one or more previous four languages.

2.1 Structure of PLC program

Start from web pages of PLCopen, which is corresponding the standardisation.

PLCopen
http://www.plcopen.org/index.html

PLCopen: IEC 61131-3: a standard programming resource
http://www.plcopen.org/pages/tc1_standards/downloads/intro_iec.pdf / Moodle material bank

Notice especially: definitions of data types, structure of configuration (how program structure is composed on the controller), and of course the definitions of five languages themselves.

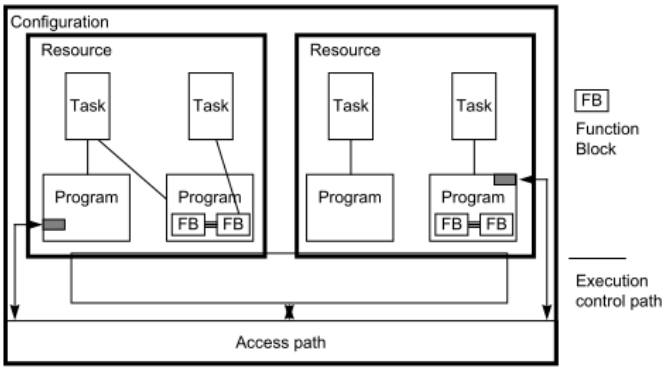

Figure 3 Architecture and structure of PLC program components

Configuration defines physical IOs, devices, memory (addresses) and it has one or more controller resources (PLC / Controller), which executes the actual program(s). The resource has 1..n *Task(s)*. Tasks are either cyclic or event based. For cyclic task is defined the frequency i.e. how often the control cycle is executed (e.g. 1ms, 5ms, 10ms, 100ms). Each Task calls (contains) 1..n Program(s). Program is composed from code and calls to other programs (PRG), Function Blocks (FB) and Function (F). Modular programs can be implemented by combining these reusable code blocks. These three kind of block entities are programmed with the previously mentioned PLC languages.

How the program should be structured? Structuring Program Development with IEC 61131-3
http://www.plcopen.org/pages/benefits/program_development/

The differences of PRG, FB and F and purpose of use of the five languages are discussed in: Beckhoff peruskurssin materiaali (Material of Beckhoff's Basic Course) **see pages 18..22 (Chapters 2.1.1 .. 2.2)**. See Moodle material bank. This material is available ONLY in Finnish. Sorry!

Beckhoff: PLC Programming Languages. Focus especially on languages SFC, ST, LD, FBD (and IL). In this order of importance!
http://infosys.beckhoff.com/content/1033/tcplccontrol/html/tcplcctrl_languages.htm?id=23675

Beckhoff: Sample Program (may skip at this stage)
http://infosys.beckhoff.com/content/1033/tcplccontrol/html/tcplcctrl_sample.htm?id=23693

2.2 PLC programming tutorials

2.2.1 SFC programming

TUT SFC introduction (Spoken in Finnish)
https://www.youtube.com/watch?v=i_aKJ6DumDI&index=4&list=PLqXlLCsUe6g6rHhtgkEMnkl7UfBbWpU3J

TUT SFC programming example (Spoken in Finnish)
https://www.youtube.com/watch?v=xNM20sHBGgY&index=5&list=PLqXlLCsUe6g6rHhtgkEMnkl7UfBbWpU3J

2.2.2 FBD programming

PLC Programming Tutorial for Beginners_ Part 2. Siemens PLC and FBD programming. A mixer tank is used as example process.
https://www.youtube.com/watch?v=nYr8Q21nG0k

2.2.3 LD programming

PLCeUniversity PLCProfessor youtube videos:
https://www.youtube.com/user/plcprofessor/videos?sort=dd&view=0&shelf_id=1

Program a PLC: Prog-1a..Prog-1f. I think good and worth watching are 1a and 1e. Direct links given in the following.

2.2.3.1 Program a PLC: Prog-1a

Prog-1a How to Program a PLC Introduction - Basic Level. Programming example of garage door. Programming language LD. Start from time 6:21. The beginning is just for introduction of case example door…
https://www.youtube.com/watch?v=srFbEVCa1T8

2.2.3.2 Program a PLC: Prog-1e

Prog-1e How To Program a PLC Introduction - Advanced Level 1. Programming example of garage door. Controls with single push button.
https://www.youtube.com/watch?v=n5MUDgLVwMQ

2.3 Other tutorials

PLC tutorial
http://www.machine-information-systems.com/Free_PLC_Tutorial.html

PLC Manual
http://www.plcmanual.com/

3 PLC Program Libraries

OSCAT library / Open Source Community for Automation Technology

http://www.oscat.de/

Library contains ready codes and functions → For Codesys only?

4 Applying and Best Practices

4.1 IPC-SMEMA

Mechanical Equipment - Interface Standard - IPC-SMEMA-9851 / See Moodle material package

http://www.ipc.org/ContentPage.aspx?pageid=IPC-SMEMA-Council-Relevant-Standards
http://www.ipc.org/html/IPC-SMEMA-9851.pdf

http://ocmmanufacturing.com/resources/resource_docs/IPC-SMEMA-9851.pdf

See Ch4 → Especially Fig4-1, Table 4-2, Table4-3, Fig 4-1, and Fig 4-5

4.2 OMAC PackML

OMAC PackML = ISA-TR88.00.02-2015, Machine and Unit States: An implementation example of ANSI/ISA-88.00.01 / See moodle material package

http://www.omac.org/content/packaging-workgroup
http://omac.org/sites/default/files/OMACandPackML_MettlerToledo.pdf
https://en.wikipedia.org/wiki/PackML
https://www.isa.org/store/isa-tr880002-2015,-machine-and-unit-states-an-implementation-example-of-ansi/isa-880001-/43761120

5 References

1. Sparx Systems / Eneterprise Architect SW. UML 2 State Machine Diagram. https://sparxsystems.com/resources/tutorials/uml2/state-diagram.html