CDL-PLC translator: From Modelica HVAC control design to IEC 61131 PLC implementation

Karl Walther[1,2] Michael Wetter[3] Anand Prakash[3] Jianjun Hu[3]

[1]Department of Mechanical Engineering, KU Leuven, Leuven, Belgium, karl.walther@kuleuven.be
[2]School of Architecture and Civil Engineering, University of Wuppertal, Wuppertal, Germany
[3]Lawrence Berkeley National Laboratory, Berkeley, CA, USA

Abstract

This paper presents the CDL-PLC translator, a tool to convert control sequences for building heating, ventilation, and air conditioning (HVAC) systems expressed in a subset of Modelica called Control Description Language (CDL) which is undergoing standardization via the voluntary ASHRAE Standard 231P into the IEC 61131-10 XML format. Such a translation is a step in a model-based design workflow and contributes to digitalizing the control delivery process in the building sector. The translation into the IEC 61131-10 XML is the last step in the implementation of control logic developed in CDL on programmable logic controllers (PLCs) standardized in IEC 61131. The paper presents the details of the translator, an example for the validation of a CDL block vs. the corresponding IEC block, and a practical application example for the translation of a control sequence for a hybrid heat pump plant from a case study neighborhood in Belgium. Practical applications, use cases, and future developments are discussed in the context of building industry processes.

Keywords: HVAC systems, Control Description Language, PLCs, Model-based Design

1 Introduction

1.1 Motivation

The building sector is facing major challenges to improve energy efficiency and load flexibility. In order to increase the share of renewable and residual energy sources, new supply systems will be installed in new and existing buildings the next years. The integration of intermittent renewable energies raises the complexity of design and operation as modern installations often use multiple technologies, such as heat pumps, solar systems and storage in order to meet new requirements for buildings to shift supply and demand and to interact with electrical grid signals. Optimizing controls of air handling units and room-side emission systems is another strategy to increase the efficiency of new and existing HVAC systems in buildings. To perform these engineering tasks in building practice, the methods, tools, and procedures for the development of HVAC control sequences come into focus.

1.2 Industry practice and related problems

In the buildings industry, the initial development and the final implementation of HVAC system controls are separated between design engineers and installation contractors (see 'Building practice' in Figure 1). This is a structural difference to, for example, the manufacturing or automotive industry. The design and the execution phases are separated by the tendering and award procedure.

The design and the development of HVAC control sequences is largely based on static considerations in single operation modes, rules of thumb, and often based on previous projects. Control sequences are documented using graphical control schemes and textual so-called 'functional descriptions' or 'sequence specifications', for example based on VDI 3814-4.3:2022 (Verein Deutscher Ingenieure 2022). These analog documents are then manually interpreted by the control provider who develops control logic for the building automation system. This practice is highly susceptible to human error (Barwig et al. 2002; Torabi et al. 2022), and leads to several well-documented problems, such as inappropriately parameterized control sequences (Crowe et al. 2020; Raftery et al. 2024) and a gap between predicted and measured performance (Wilde 2014). An in-depth revision of the control sequences in the control code, which goes beyond parameter optimization, requires qualified engineers and is therefore costly.

1.3 Model-based HVAC control design

Model-based design is an approach to master the increasing complexity of future HVAC systems and their controls through an increased level of detail and increased quality in the design phase. Based on regulatory or user-defined requirements for the energy and comfort performance, a model-based and digitalized workflow generally includes three parts (see dashed boxes in Figure 1):

1. A Building Performance Simulation (BPS) environment for the integrated simulation of the building envelope, HVAC systems, and their controls, and the verification against performance requirements.

2. A digital exchange format for the lossless and unambiguous documentation and transfer of control sequences.

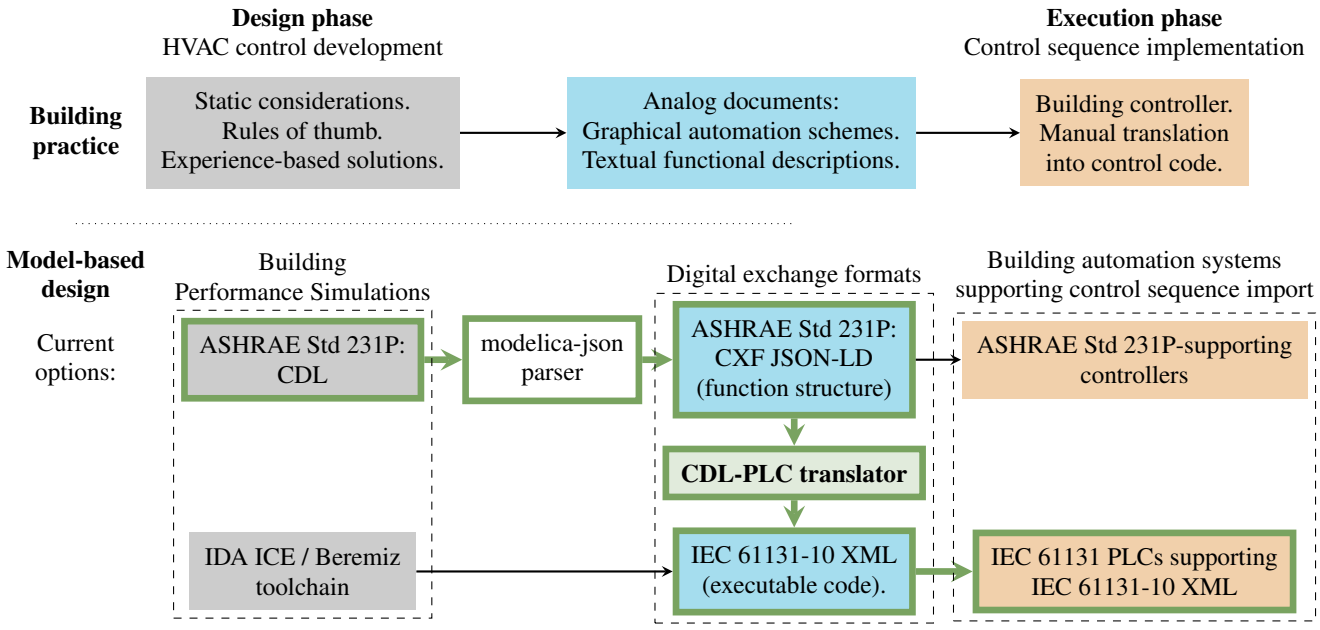


Figure 1. CDL-PLC translator in the context of development and implementation of HVAC controls in building practice, model-based design, and the workflow described in this study (green label). Note that from CDL to CXF, there is an alternate path via the ctrl-flow software[1] that does not require a building performance simulation. This path is not shown here as it does not include performance evaluation as is typical in model-based design.

3. For a seamless and efficient implementation, a building automation system (BAS) supporting the import of control sequences is needed.

Model-based design of HVAC controls is a new approach in the building industry, and generally applicable to all types of buildings. However, due to the effort required for model development, complex and individual HVAC systems in commercial buildings and supply systems in urban districts and neighborhoods would be typical applications. While model-based design is generally independent of the control approach, the toolchains discussed in this paper use conventional Rule-Based Controls (RBCs), which are still the standard in the building sector.

In the literature, two approaches can be distinguished that are presented in the following two sections.

IEC 61131-3 code simulation in IDA ICE The first option (see bottom row in Figure 1) uses the IEC 61131-3 code simulation in the commercial equation-based BPS software IDA ICE, the IEC 61131-10 XML standard as digital exchange format, and IEC 61131 PLCs as target system. The idea behind this toolchain is to simulate the behavior of real controllers in an equation-based simulation environment with a variable step size solver, such as IDA ICE (Sahlin, Bring, and Eriksson 2009). This toolchain was initially developed for Software-In-the-Loop (SIL) testing of PLC programs during the commissioning phase (Sahlin, Skogqvist, and Högberg 2018), however, it can also be used for the development of HVAC controls during the design phase, as demonstrated in a full-scale end-to-end application for air handling units (AHUs) in a factory building (Walther 2025).

IEC 61131-3 (International Electrotechnical Commission 2003) is a standard for the programming of controllers standardized in IEC 61131, called PLCs. IEC 61131 PLCs are commonly used for building automation in Europe. PLC programs are organized in program organization units (POUs) that can be a type or an instance of functions, function blocks, or programs. IEC 61131-3 defines 5 languages, two of which are textual (Structured Text (ST), Instruction List (IL)) and three of which are graphical (Function Block Diagram (FBD), ST, IL), which can be different for each POU. In chapter 2.5.1.5. of IEC 61131-3, standard functions for arithmetic, numerical, type conversion operations etc. are defined. For a technology overview and suggestions for future developments, e.g. to achieve deterministic behavior, we refer to Sehr et al. (2021).

In the IDA ICE toolchain, an XML file standardized in IEC 61131-10 is used for the digital delivery of control sequences from the design to the execution phase. The first version of this XML format was published in 2005 under the umbrella of the PLCopen organization (Marcos et al. 2009). IEC 61131-10 XML was originally designed to exchange control sequences between different PLC Integrated Development Environments (IDEs). Some PLC IDEs support the IEC 61131-10 XML import, for example Codesys, TwinCAT (Beckhoff), or e!cockpit (Wago). However, due to the above mentioned construction-specific workflow procedures (no code development in the design phase, normatively prescribed graphical and textual exchange formats), it is not used in

[1] See `https://ctrl-flow.lbl.gov`.

DOI 10.3384/ecp218515

the building sector today.

Modelica Control Description Language Within the OpenBuildingControl (OBC) project, a new process and supporting tools was developed to digitalize specification, implementation, and verification of HVAC control sequences (Wetter, Ehrlich, et al. 2022) (see middle row in Figure 1). The central part of CDL[2] is a library with standardized function blocks typically used for building applications, as a small subset of Modelica (Wetter, Grahovac, and Hu 2018a). This library alongside with rules for how to compose control logic in a hierarchical way was first presented in (Wetter, Grahovac, and Hu 2018b) and it is currently being standardized in ASHRAE Standard 231P.

To transfer HVAC control sequences defined in Modelica, multiple options are possible.[3] In principle, the (already existing) Functional Mock-up Interface (FMI) standard can be used, however, today's Building Automation Systems mostly do not support the direct execution of Functional Mock-up Units (FMUs) / C code, and they do not support editing Modelica models and regenerating FMUs if the control logic need to be changed after installation. To address this issue, the modelica-json translator[4] was developed to parse CDL sequences into an intermediate JSON file format, called Control Exchange Format (CXF) (Wetter, Hu, et al. 2021). With such an intermediate format, the translation of control sequences developed in CDL into product-line specific languages in general is performed in two steps:

1. *Parsing* control sequences in CDL into a JSON CXF using the open-source modelica-json translator. This step is fully covered by the CDL toolchain (open-source).

2. *Translation* of the CDL JSON into the product-line specific language. If the target system uses proprietary language, this step has to be performed by the controller manufacturer.

CDL vs. IEC 61131 The previously described options have introduced two standards, IEC 61131 and CDL (ASHRAE Standard 231P). Different areas of application are briefly contrasted below to underline that the two standards are not mutually exclusive.

IEC 61131, on the one hand, is a standard for controllers. It standardizes a computational model, programming languages, interfaces, exchange formats, etc. The IEC 61131 standard is only adopted by some manufacturers, while other manufacturers use proprietary control implementations. The crucial step that influences the actual building performance in operation most is, however, the early design phase when control sequences have to be designed in the building context, and when the actual control vendor might not yet be decided.[5] CDL, on the other hand, is specifically designed to allow simulation of control sequences together with HVAC systems in the building context, and subsequent translation to a Building Automation System. CDL is product-neutral and addresses all downstream controller types, including those that follow IEC 61131.

A further crucial aspect is that CDL provides a library of standardized function blocks. Such a library is missing in the PLC-domain where usually each vendor provides proprietary libraries. These vendor-specific libraries have several drawbacks: (a) they are typically proprietary and only documented textually, (b) they are not standardized and vary from vendor to vendor which locks customers into specific systems, and (c), most importantly, they only come into play during the programming of controllers during the execution phase and are therefore inevitably differ from what mechanical engineers have drafted during the design stage.

1.4 Research gap and contribution

In the context of CDL, current efforts focus on the standardization of control blocks and the exchange formats, such as the CXF JSON-LD, in ASHRAE Standard 231P (first step, 'parsing', described in 'Modelica-based Control Description Language' above).

This study presents the CDL-PLC translator to address the final step of transferring CDL sequences to building controllers following IEC 61131. The translator converts the CXF JSON into the IEC 61131-10 XML, and as such link CDL and the PLC-domain (see green path in Figure 1). The particular benefit maximizing the impact of such a translator is that CDL is linked to IEC 61131 which is followed by several automation manufacturers.

The translation of the CDL CXF into the IEC 61131-10 XML is associated with two main challenges:

1. The *translation of the structure* of the CXF JSON-LD into the target structure, here the IEC 61131-10 XML.

2. The *implementation of control functions* defined in CDL blocks into the language of the target system, here control code according to IEC 61131-3.

The remainder of this paper is organized as follows: First, the CDL-PLC translator is presented in section 2. In section 3, the translator is applied to a control sequence from a real project to demonstrate the functionality in a practical case. We conclude with a discussion on industry transformation and future work.

2 CDL-PLC translator

In this section, first, the general approach of the CDL-PLC translator is presented, then, source and target formats are

[2] https://obc.lbl.gov/specification/cdl.html

[3] https://obc.lbl.gov/specification/codeGeneration.html

[4] https://github.com/lbl-srg/modelica-json

[5] This applies for the case of new buildings. In existing buildings, the implemented BAS might or might not follow IEC 61131.

described, and, finally, the program architecture and the features are described.

2.1 Basics

General approach The general approach of the CDL-PLC translator is to convert the information from the CDL CXF JSON-LD format to the IEC 61131-10 XML format. The translator directly maps CDL blocks to IEC blocks, and the output IEC 61131-10 XML uses the graphical FBD language. This enables a direct and visual comparison of the graphical FBD representations in Modelica and IEC 61131-3. The translator is implemented in Python. For all PLC-related tasks within this study, the open-source PLC IDE Beremiz[6] was used.

Workflow and use cases The CDL-PLC translator is meant to be used either by design engineers or by executing contractors depending on how the control delivery is organized. During the execution phase, automation contractors tasked with the programming of IEC 61131 PLCs then import the IEC 61131-10 XML into the product-line specific PLC IDE.

The translator addresses two building-specific use cases:

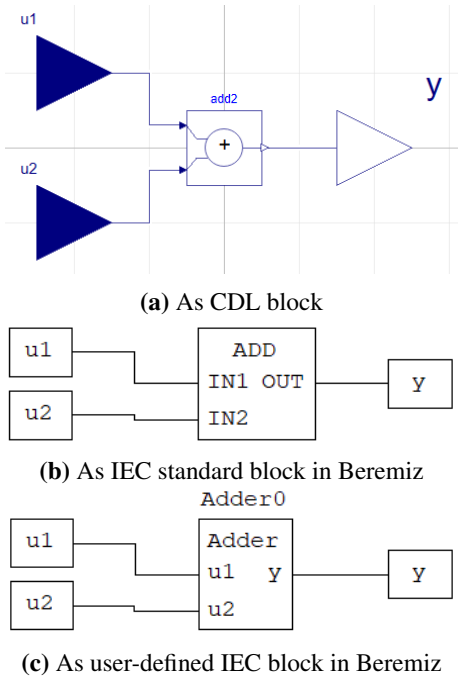
1. The development of control sequences in *new buildings*. In this case, the option to import the IEC 61131-10 XML must be specified in the tender. If a BAS already supports the FMI standard, importing an FMU might be a more straightforward option.

2. The revision of control sequences in *existing buildings*, where the existing BAS is kept, and where an existing HVAC system is either kept or replaced within a renovation. Only BAS with PLCs following IEC 61131 and IDEs supporting the import of the IEC 61131-10 XML are discussed below. As new PLC product lines might directly support the FMI standard (see section 4), such a redesign of existing legacy PLC-based BAS is a likely application for the CDL-PLC translator.

Development status The CDL-PLC translator is currently at a prototypical stage as a proof of concept that CXF can be translated into IEC 61131-10 XML. The translator supports the necessary basic features such as mapping and connecting function blocks. The translator currently supports a small subset of CDL elementary blocks, mainly those needed for the application example presented in section 3. However, as the core features are covered, extending the translator to a fully-fledged software is straightforward.

In its current form, the translator works only in the CDL-to-PLC direction. The PLC-to-Modelica direction may be possible for some control logic and would be beneficial, for example, in case executing contractors have made changes to the PLC code that have to be verified in Modelica, or when operational problems are easier to be addressed using simulation. However, whether a translation from a Building Automation System to CDL is possible is case dependent, as Building Automation Systems allow for formulations that are outside of the allowed formulation of Modelica or its CDL subset. For example, Wetter, Chen, et al. (2023) shows some differences between how control logic is implemented in building automation systems and in Modelica.

[6] https://beremiz.org/



(a) As CDL block

(b) As IEC standard block in Beremiz

(c) As user-defined IEC block in Beremiz

Figure 2. Control sequence with an adder as FBD

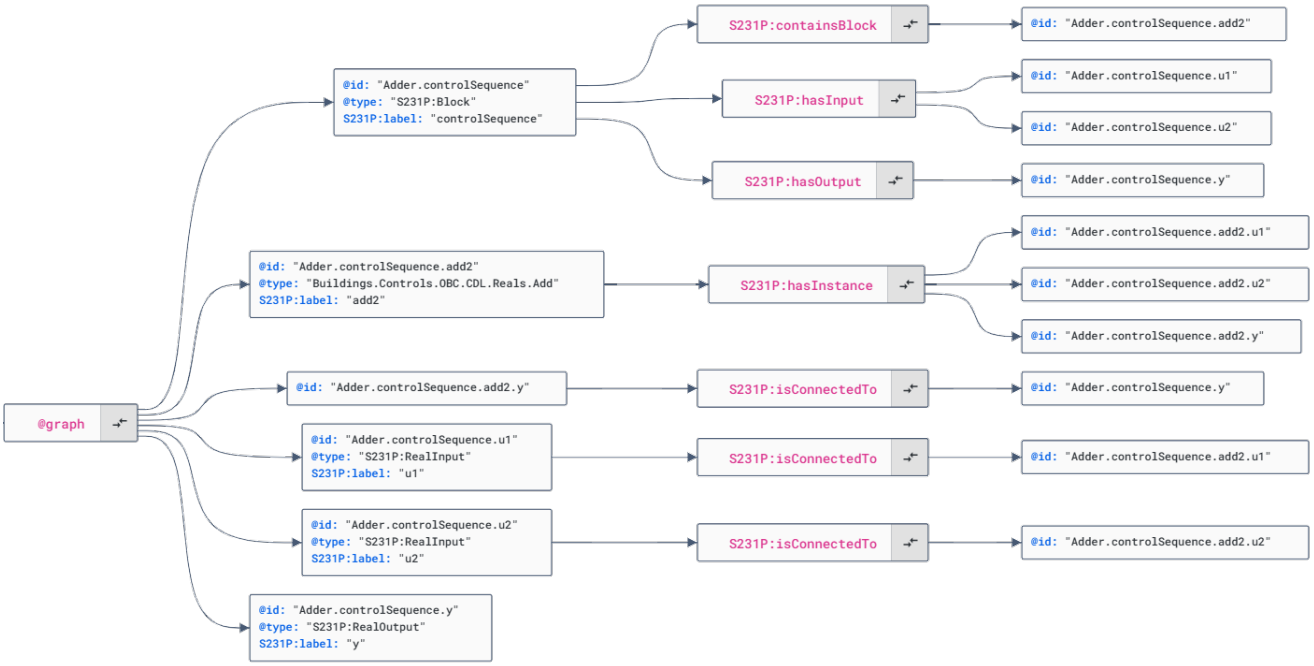
2.2 Source and target formats

In the following two sections, the CXF JSON-LD source format and the IEC 61131-10 XML target format are described in order to outline differences and the functional prerequisites for the translator. The description is done for a simple control sequence with inputs `u1` and `u2`, an adder block, and output `y` as shown in Figure 2.

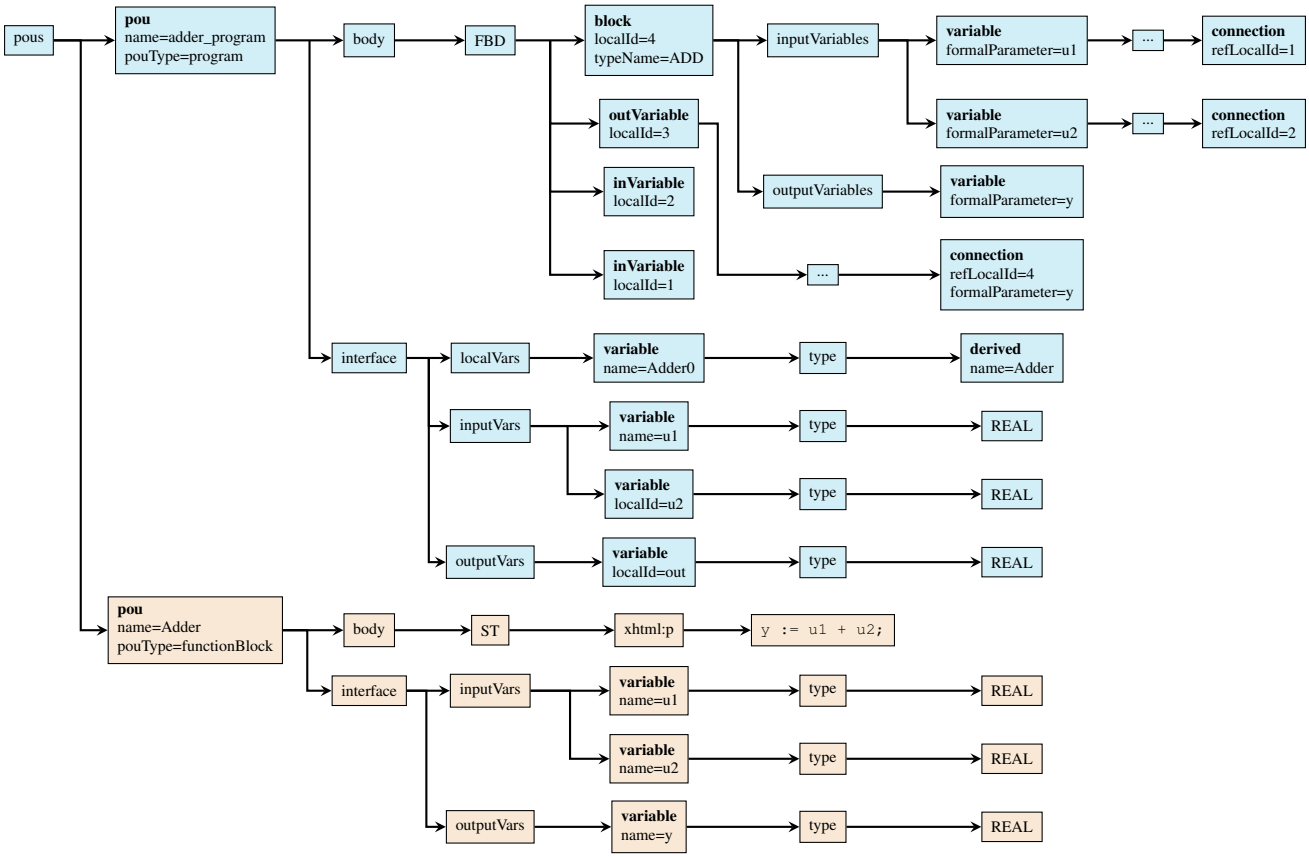
2.2.1 CXF JSON-LD

In CXF, nodes are separated depending on whether they represent composite or extended blocks, inputs (`Real`, `Integer`, or `Boolean`), outputs (`Real`, `Integer`, or `Boolean`), elementary blocks, parameter definitions, parameter assignments, or connections. Figure 2a depicts a simple control sequence with the instance `add1` of `Buildings.Controls.OBC.CDL.Reals.Add`. The visualization in Figure 3a shows the graph objects: the control sequence itself (`@id: "Adder.controlSequence"`), inputs, outputs, the adder block instance, and the output `y` of `add1`.

DOI
10.3384/ecp218515



(a) CXF JSON-LD



(b) IEC 61131-10 XML. Top branch (blue): IEC Program. Bottom branch (orange): IEC function block declaration.

Figure 3. Visualized graphs of the example from Figure 2 (in both visualizations, information, such as for the graphical rendering, is removed to improve readability).

2.2.2 IEC 61131-10 XML

Figure 2b depicts the control sequence as FBD in Beremiz, Figure 3b depicts the structure of the corresponding IEC 61131-10 XML. In Figure 3b, for example, the instance `adder_program` is of `pouType=program` and the instance `Adder` is of `pouType=functionBlock`. While the instance `program` uses the `FBD` language (IEC Function Block Diagram), the instance `adder` is expressed in `ST` (IEC Structured Text). Figure 2b and Figure 2c also illustrate the difference between an IEC 61131-3 standard adder (see Figure 2b) and a user-defined function blocks (see Figure 2c, where the output in the `Adder` class is defined as `y := u1 + u2;`). Connections between block instances and inputs and outputs (IOs) are described in an *input-is-linked-to-output* form. IDs assigned to each block instance, each input variable, and each output variable (see `localId` attribute), are used as identifiers for connections.

2.3 Program flow

The translation is executed in two steps that are explained in the following two sections.

2.3.1 Information extraction and conversion

In a first step, the information included in the CXF JSON-LD is extracted and converted into Python objects. According to the structure of the IEC XML, the objects are grouped as follows:

- Inputs
- Outputs
- Parameters (see next paragraph)
- Blocks (with/without direct CDL-IEC equivalent separated, see next paragraph)
- Composite blocks

Handling of parameters IEC 61131-3 does not support the concept of parameters editable per function block instance as in Modelica or its CDL subset. As a workaround, CDL parameters are converted into program inputs assigned to the corresponding function block instance (see application example in subsection 3.3). The translator also supports global parameters that can be propagated to one or more block instances. An option to avoid additional program inputs would be using IEC 61131-3 *local variables*. However, such *local variables* are fixed per class declaration rather than class instance, which would require creating different class declarations for each set of parameters.

Mapping of CDL elementary blocks and validation CDL blocks are mapped to the equivalent blocks in PLC code as follows:

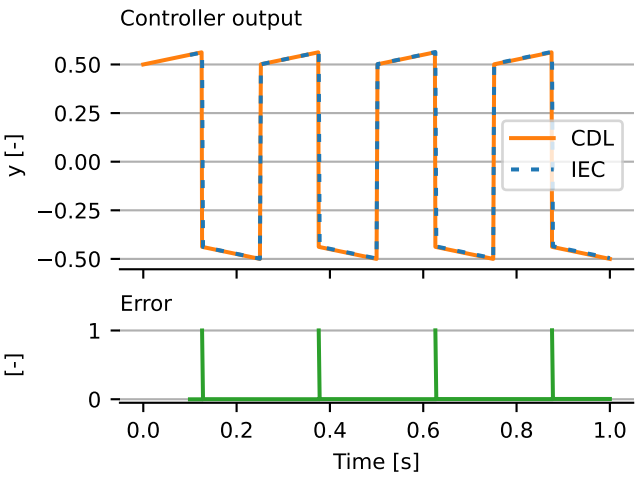


Figure 4. Validation of the `Reals.PID` CDL block vs. the corresponding block translated into IEC 61131-3 code.

- If possible, CDL blocks are directly mapped to the corresponding *IEC standard blocks* (see subsubsection 2.2.2). For example, `Logical.And` (CDL) is mapped to `AND` (IEC 61131-3).
- If a direct mapping is not possible, CDL function blocks are *individually translated* into IEC code and stored as XML snippets (see for example `Adder` function block class in Figure 3b).

An approach to facilitate the translation of each block individually would be using PLC function blocks from the open-source OSCAT library[7]. However, OSCAT blocks often do not fit exactly to the CDL definitions. OSCAT blocks are, however, used to process the PLC time (see section 'Different implementations of IEC 61131-3'), and for the integration block within the PID block.

To check whether the CDL elementary blocks have been correctly translated into the target language, the CDL toolchain offers validation packages. Figure 4 shows the result of such a validation for the `Reals.PID` block with a simulation time of 1 s. For the validation, a pulse input for the setpoint with a period time of 250 ms is prescribed on the PI controller setpoint u_s (fixed measurement input $u_m = 0.5$, proportional gain $k = 1$, time constant of the integrator block $T_i = 1$. The controller output of the IEC block matches with the CDL trajectory with a constant error of $e = 0.004$. The peaks occur because the IEC controller (cycle time 2 ms) switches one tick later than the CDL controller.

Translation of topology While connections are an attribute of output connectors in the CXF, they are an attribute of inputs in the IEC XML. The translator performs the necessary reassignment.

[7] `http://www.oscat.de`

DOI 10.3384/ecp218515

Listing 1. XML template for a function block instance

```
<block localId="{{ values.localId }}"
    typeName="{{ values.ClassName}}"
    instanceName="{{ key }}"
    executionOrderId="0" width="{{ values.
    width }}" height="{{ values.height }}">
        <position x="{{ values.x_absolute
            }}" y="{{ values.y_absolute }}"
            />
        <inputVariables>
                {% for key, value in values
                    .inputs.items() %}{%
                    include "xml_templates/
                    structure/
                    FbdBlockInstanceInVar.
                    xml" %}{% endfor %}
        </inputVariables>
        <inOutVariables/>
        <outputVariables>
                {% for key, value in values
                    .outputs.items() %}{%
                    include "xml_templates/
                    structure/
                    FbdBlockInstanceOutVar.
                    xml" %}{% endfor %}
        </outputVariables>
</block>
```

Different implementations of IEC 61131-3 Despite the standardization in IEC 61131, there are differences in the implementation of the standard between the control manufacturers, e.g., in the way the control time is updated, e.g. for the integrator in PID controllers. This requires that different functions / function blocks have to be mapped depending on the specific IEC 61131 target system. The CDL-PLC translator is currently outputting an IEC XML that is tested in the open-source PLC IDE Beremiz. Further adaptations to the XML might be required depending on the target IDE (not investigated in this study). In the application described in Walther (2025) an XML created with Beremiz is imported into the commercial PLC IDE e!cockpit.

2.3.2 Rendering

In a second step, the information collected in the Python objects is rendered in a IEC 61131-10 XML. For this rendering, the translator iterates over each level of the XML tree, and fills the information included in the corresponding Python objects in predefined XML templates. As an example, Listing 1 shows the template for the function block instance level of a `program` POU (cf. Figure 3b). For the rendering, the Python package 'Jinja2'[8] is used.

3 Application example

In this section, an application example is presented to highlight the currently developed features of the translator.

[8] `https://pypi.org/project/Jinja2/`

3.1 Demonstration building and HVAC system

CDL-PLC is used to translate a control sequence of a heat pump-based supply system for the renewable and residual energy sources (R^2ES)-based renovation of the 'Stijn Streuvelstraat' neighborhood in Bruges, Belgium. The project details are described in Walther et al. (2025).

3.2 Description of the control sequence

The control sequence is used to regulate a hybrid heating supply system with an air-source heat pump (ASHP) and a ground-source heat pump (GSHP). The FBD representation in Modelica is depicted in Figure 5. The control sequence has two inputs (the outdoor temperature `TAmb` and the tank temperature `TTan`) and two outputs (the two control signals `yAsHp` and `yGsHp` for the modulation of the two heat pumps, assuming that the compressor speed can be directly controlled). A so-called 'sequence controller' with a PI controller and a signal splitter is used to initially modulate only one heat pump, and to add the second heat pump if required. The `signalSplitter` block splits the input signal between `0` and `1` into two sequential output signals such that the first output increases from `0` to `1` when the input is between `0` and `0.5`, and the second output increases accordingly from `0` to `1` for inputs between `0.5` and `1`.

The hysteresis instance `hys` is used to switch the priority depending on the outdoor temperature: at lower outdoor temperatures the GSHP is the first generator in the sequence, and at higher outdoor temperatures the ASHP.

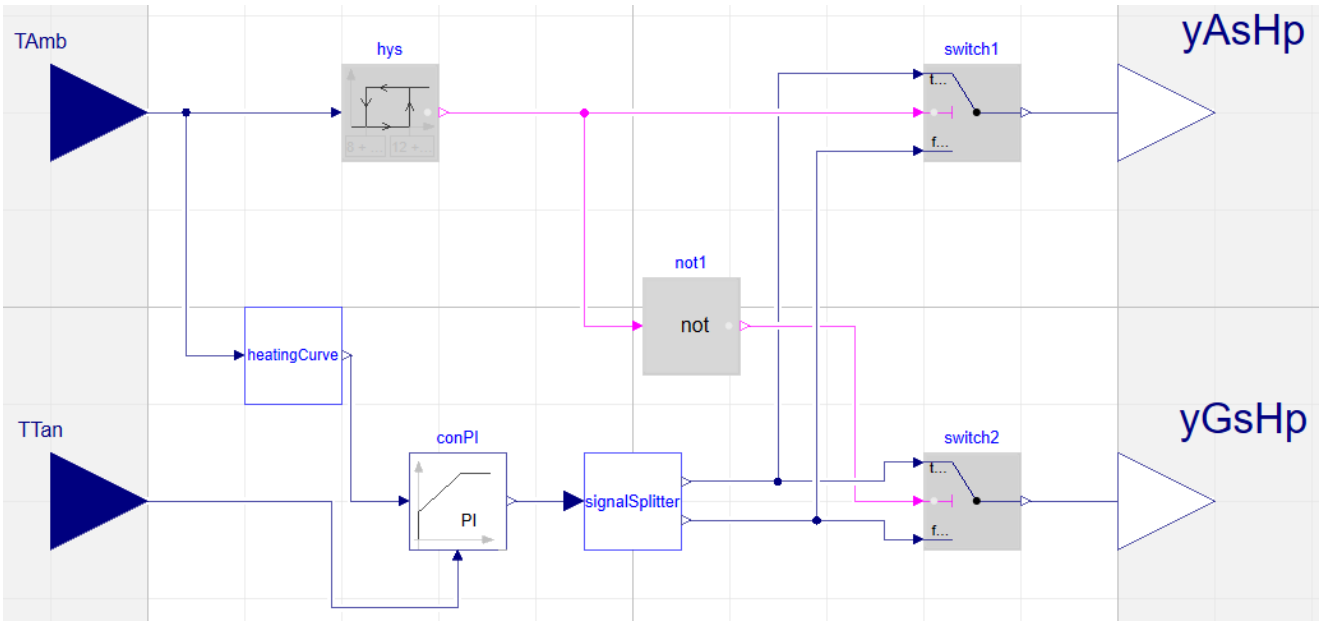
3.3 Comparison of CXF JSON-LD and IEC 61131-10 XML

The CDL source sequence and the translated IEC 61131-3 sequence are depicted in Figure 5 and Figure 5b.

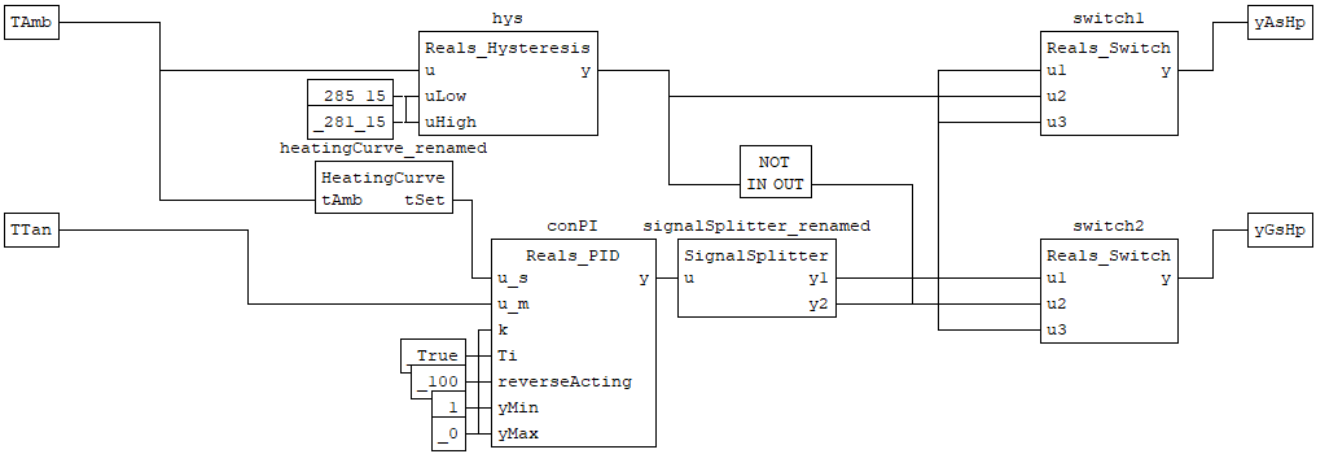
Mapping of inputs, outputs, function blocks, and connections Figure 5b shows that the two inputs (`TAmb` and `TTan`) and the two outputs (`yAsHp` and `yGsHp`) have been transferred accordingly from the CDL representation (Figure 5). This illustrates that the definition of identifiers for inputs and outputs is already done during the design phase in Modelica, and not, as is current practice in construction, only during the commissioning phase.

As described in subsubsection 2.3.1, two cases can be distinguished: (1) CDL blocks that have a direct equivalent in IEC 61131-3 standard functions are mapped to those: the CDL elementary block `not1` (type `Logical.Not`) is mapped to the IEC 61131-3 standard block `NOT`. (2) CDL blocks without a direct equivalent in IEC 61131-3 standard functions, such as `Reals.Hysteresis` or `Reals.PID`, are mapped to the individually translated IEC 61131-3 blocks.

Note that for `Reals.PID`, which is an elementary block, we translated it as if it were a composite block because the CDL implementation is in fact done using only constructs as allowed for composite blocks. In this example, `conPI` is configured as PI controller, which can



(a) Graphical view of CDL, as rendered by Dymola



(b) Translation in PLC code (Beremiz)

Figure 5. Comparison of function block diagrams for a heat pump control sequence

be defined via the `controllerType` parameter. The two user-defined composite blocks `heatingCurve` and `signalSplitter` are translated recursively by iterating through the hierarchical instance tree.

Figure 5b also shows that connections are correctly translated.

Parameter propagation Figure 5b illustrates how parameters of the `hys` and the `conPI` CDL blocks are transformed into IEC program inputs that are connected to the corresponding block inputs. The parameter value with a leading underscore is used as name for the input object.

Graphical representation The comparison of Figure 5 and Figure 5b reveals differences in the graphical rendering of the FBD representation. To take into account the variable size of function blocks in IEC FBDs and avoid overlapping blocks, the IEC FBD layout is stretched in x and y directions. This is, however, a particular issue of Beremiz, where blocks can be positioned freely. Other (commercial) IDEs, such as Codesys or TwinCAT, ignore the x-y-coordinate information and re-position the blocks depending on the relation on inputs and outputs.

4 Conclusion and future work

This study has presented the CDL-PLC translator for the conversion of Rule-Based Control sequences from the Modelica-based CDL (Control Description Language) into the IEC 61131-10 XML for PLCs. The CDL-PLC translator is an example for the translation of CXF to a controller specific language, and the last step for the implementation of HVAC control sequences expressed in CDL on IEC 61131 building controllers. We presented an example for the validation of an IEC block against its

DOI
10.3384/ecp218515

CDL equivalent, and an example for the translation of a real HVAC control sequence.

Implementing model-based design poses several challenges along the building life cycle. First and foremost, developing BPS software that enables the simulation of HVAC systems and controls for buildings with up to several hundreds of zones fast enough to be usable by practitioners remains challenging (Sahlin, Skogqvist, Lebedev, et al. 2019; Wetter, Benne, et al. 2023). In the meantime, simplifying load models such as by adding multiple rooms to one representative room is a recommended work-around.[9] For mechanical / HVAC and control engineers, this transformation can be done in two ways: Either by performing detailed HVAC control sequence development, for example using Modelica editors, during the pre-award design phase, which is not done today. Alternatively, ready-to-configure control and HVAC templates can be developed by library developers as shown in Gautier et al. (2023) and distributed with Modelica libraries, possibly integrated in configuration tools such as ctrl-flow.[10] The challenge for PLC vendors is to provide control sequence import functionality, either, if not already available, for the IEC 61131-10 XML or directly for the FMI standard. A noteworthy development is that a first PLC vendor starts to support the FMI standard[11].

To further develop and establish CDL-based model-based design of HVAC systems, joint efforts of academia and industry partners are required. Joint efforts need to focus not only on software toolchains and interfaces, but also - and perhaps even more importantly - on business models and best practice workflows to show various stakeholders the path to and benefit of adopting such model-based, digitalized processes.

The CDL-PLC translator is currently at a prototypical stage supporting a few CDL blocks as a proof of concept that such a translation is feasible. For a fully-fledged software, all CDL blocks have to be mapped and translated, systematically validated against their CDL counterparts, lacking features, such as the support of arrays[12], have to be added, and the import in different, particularly commercial, PLC IDEs has to be tested. Another challenge is the implementation of state machines in the translator. In CDL, a state machine can be implemented through so-called 'extension blocks' that are translated into FMUs.[13] The implementation of FMUs on the target automation system requires, however, that the target system is FMI-compatible, while the presented translator primarily targets legacy PLC-based automation systems that do not yet support the FMI standard (see '1' in 'Workflow and use cases'). One possible workaround would be a custom translator that converts the state machine code in the extension block to equivalent PLC code.

An important aspect for automated workflows is a complementary semantic model. CDL and its CXF translation supports such a semantic model based on ASHRAE Standard 223P, however, this information is lost when translating from CXF to IEC 61131-10 XML. To link the IEC 61131-10 XML with a semantic model, Ihlenburg et al. (2024) have proposed the PLCont methodology, however, further efforts towards a full integration or an extension of the XML schema are required.

[9] The simulation time for the control sequence used for the application example in section 3 with 61 thermal zones is around 24 h (Walther et al. 2025)

[10] https://ctrl-flow.lbl.gov.

[11] https://github.com/Beckhoff/TE142x---FMU-Samples/

[12] https://obc.lbl.gov/specification/cdl.html#arrays

[13] https://obc.lbl.gov/specification/cdl.html#sec-ext-blo

Data availability

The CDL-PLC translator is available under https://github.com/lbl-srg/cdl-plc.

Acknowledgements

The authors acknowledge the funding by the European Union through the SEEDS project under the Horizon Europe Programme (grant Agreement: 101138211). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Executive Agency (REA). Neither the European Union nor the granting authority can be held responsible for them.

Funded by the European Union

This research was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Building Technologies Office, of the U.S. Department of Energy, under Contract No. DE-AC02-05CH11231.

References

Barwig, Floyd E. et al. (2002-08). "The National Building Controls Information Program". In: Summer Study on Energy Efficiency in Buildings. ACEEE. Pacific Grove, CA.

Crowe, Eliot et al. (2020). "Building commissioning costs and savings across three decades and 1500 North American buildings". In: *Energy and Buildings* 227, p. 110408. ISSN: 0378-7788. DOI: 10.1016/j.enbuild.2020.110408.

Gautier, Antoine et al. (2023-10). "HVAC and Control Templates for the Modelica Buildings Library". In: *Proceedings of the 15th International Modelica Conference*. Linköping Electronic Conference Proceedings. Aachen, Germany: Modelica Association and Linköping University Electronic Press, pp. 217–228. DOI: 10.3384/ecp204217.

Ihlenburg, Moritz et al. (2024). "Towards a Digital Representation of Building Systems Controls". In: *International Sustainable Energy Conference - Proceedings*. ISEC 2024 – 3rd International Sustainable Energy Conference. Vol. 1. Graz, Austria: AEE - Institut für Nachhaltige Technologien (AEE INTEC). DOI: 10.52825/isec.v1i.1217.

International Electrotechnical Commission (2003). *Programmable Controllers: Part 3: Programming Languages*.

Marcos, Marga et al. (2009). "XML Exchange of Control Programs". In: *IEEE Industrial Electronics Magazine* 3.4, pp. 32–35. ISSN: 1932-4529. DOI: 10.1109/MIE.2009.934794.

Raftery, Paul et al. (2024). "Insights from Hydronic Heating Systems in 259 Commercial Buildings". In: *Energy and Buildings*, p. 114543. ISSN: 03787788. DOI: 10.1016/j.enbuild.2024.114543.

Sahlin, Per, Axel Bring, and Lars Eriksson (2009). "Real Controllers in the Context of Full-Building, Whole-Year Simulation". In: *Eleventh International IBPSA Conference*. Building Simulation 2009. Glasgow, Scotland: IBPSA, pp. 72–79. URL: https://publications.ibpsa.org/conference/paper/?id=bs2009_0072_79.

Sahlin, Per, Patrik Skogqvist, and Markus Högberg (2018). *IEC 61131-3 Code Simulation for Software-in-the-loop PLC Testing with EQUA Tools*. 15 pp. URL: https://itea3.org/project/workpackage/document/download/5090/D3.5%20OPENCPS%20-%20IEC61131-3%20Code%20Simulation%20For%20SiL%20PLC%20Testing%20M36.pdf.

Sahlin, Per, Patrik Skogqvist, Alexey Lebedev, et al. (2019). "On the Scalability of Equation-Based Building and District Simulation Models". In: *Proceedings of Building Simulation 2019: 16th Conference of IBPSA*. Ed. by Vincenzo Corrado et al. Building Simulation Conference Proceedings. IBPSA, pp. 2584–2590. DOI: 10.26868/25222708.2019.210130.

Sehr, Martin A. et al. (2021). "Programmable Logic Controllers in the Context of Industry 4.0". In: *IEEE Transactions on Industrial Informatics* 17.5, pp. 3523–3533. ISSN: 1551-3203. DOI: 10.1109/TII.2020.3007764.

Torabi, Narges et al. (2022). "Common Human Errors in Design, Installation, and Operation of VAV AHU Control Systems – A Review and a Practitioner Interview". In: *Building and Environment* 221, p. 109333. ISSN: 03601323. DOI: 10.1016/j.buildenv.2022.109333.

Verein Deutscher Ingenieure (2022). *Building Automation and Control Systems (BACS). Methods and Tools for Planning, Building, and Acceptance Tests. BACS Automation Scheme, BACS Function List, BACS Function Description*.

Walther, Karl (2025). "Model-based design, digital delivery, and implementation of HVAC controls — Lessons learned from a building-scale application for AHUs and PLCs". In: *Building and Environment* 282, p. 113126. ISSN: 03601323. DOI: 10.1016/j.buildenv.2025.113126.

Walther, Karl et al. (2025). "Simplifying Decision-Making in the Model-Based Co-Design of Building Energy Systems through Automatically Generated Optimal Controllers". In: Building Simulation 2025. Brisbane, Australia. DOI: https://doi.org/10.26868/25222708.2025.1690.

Wetter, Michael, Kyle Benne, et al. (2023). "Spawn: Coupling Modelica Buildings Library and EnergyPlus to Enable New Energy System and Control Applications". In: *Journal of Building Performance Simulation*, pp. 1–19. ISSN: 1940-1493. DOI: 10.1080/19401493.2023.2266414.

Wetter, Michael, Yan Chen, et al. (2023-09). "Bridging the gap between building energy modeling and controls implementation - experiences of, and best practice for, coupled HVAC-control modeling". In: *18-th IBPSA Conference*. Vol. 18. International Building Performance Simulation Association. Shanghai, China, pp. 791–798. DOI: 10.26868/25222708.2023.1258.

Wetter, Michael, Paul Ehrlich, et al. (2022). "OpenBuildingControl: Digitizing the Control Delivery from Building Energy Modeling to Specification, Implementation and Formal Verification". In: *Energy* 238, p. 121501. ISSN: 03605442. DOI: 10.1016/j.energy.2021.121501.

Wetter, Michael, Milica Grahovac, and Jianjun Hu (2018a). "Control Description Language". In: *Proceedings of the 1st American Modelica Conference*. Cambridge, MA: Modelica Association and Linköping University Electronic Press, pp. 17–26. ISBN: 978-91-7685-148-7. DOI: 10.3384/ECP1815417.

Wetter, Michael, Milica Grahovac, and Jianjun Hu (2018b-08). "Control Description Language". In: *1st American Modelica Conference*. Cambridge, MA, USA. DOI: 10.3384/ecp1815417.

Wetter, Michael, Jianjun Hu, et al. (2021). "Modelica-Json: Transforming Energy Models to Digitize the Control Delivery Process". In: *Proceedings of Building Simulation 2021a: 17th Conference of IBPSA*. Building Simulation Conference Proceedings. Bruges. Belgium: IBPSA, pp. 1–8. DOI: 10.26868/25222708.2021.30141.

Wilde, Pieter de (2014). "The Gap between Predicted and Measured Energy Performance of Buildings: A Framework for Investigation". In: *Automation in Construction* 41, pp. 40–49. ISSN: 09265805. DOI: 10.1016/j.autcon.2014.02.009.