User-guided engineering of integrated energy management functions in automation systems

Jan-Niklas Puls
University of Applied Sciences and Arts Faculty I – Electrical and Computer Engineering
Hannover, Germany
jan-niklas.puls@hs-hannover.de

Maxim Runge
University of Applied Sciences and Arts Faculty I – Electrical and Computer Engineering
Hannover, Germany
maxim.runge@hs-hannover.de

Prof. Dr. Karl-Heinz Niemann
University of Applied Sciences and Arts Faculty I – Electrical and Computer Engineering
Hannover, Germany
karl-heinz.niemann@hs-hannover.de

Christian Meyer
Dietz Automation & Umwelttechnik GmbH
Neukirchen-Riebelsdorf, Germany
christian.meyer@dietz-automation.de

***Abstract*—In the area of manufacturing and process automation in industrial applications, technical energy management systems are mainly used to measure, collect, store, analyze and display energy data. In addition, PLC programs on the control level are required to obtain the energy data from the field level. If the measured data is available in a PLC as a raw value, it still has to be processed by the PLC, so that it can be passed on to the higher layers in a suitable format, e.g. via OPC UA. In plants with heterogeneous field device installations, a high engineering effort is required for the creation of corresponding PLC programs. This paper describes a concept for a code generator that can be used to reduce this engineering effort.**

Keywords—technical energy management, energy efficiency, code generation, measurement data acquisition, energy monitoring

I. Introduction

Industrial production companies strive to reduce the energy consumption of their production facilities due to CO_2 reduction targets and due to cost issues [1]. One basic measure to achieve this goal, is the implementation of an energy management system according to ISO 50001 [2]. This standard defines the organizational measures needed and uses a continuous improvement process (plan, do, check, act) to achieve the goal of energy saving and cost saving. Besides the implementation of organizational measures, the ISO 50001 demands for the implementation of a technical energy management system (tEnMS) [2]. A tEnMS measures, collects, stores, displays and evaluates energy data, so that energy flows in the production plant can be continuously monitored. In addition, a tEnMS can also provide functions for load management [3]. Energy cost of industrial users depends partly on the peak load of the plant. Load management tries to flatten the load curve and to avoid peak load situations.

A tEnMS can be implemented in two ways:

1. A tEnMS can be set up in parallel to the automation system. In this case an additional infrastructure with sensors, data processing (PLCs or edge devices) and communication infrastructure is needed. This concept yields the benefit that the control system remains untouched, in case a tEnMS is implemented after the commissioning of the control system.
2. A tEnMS can be integrated into the existing structure of the automation system. Since already existing components (controllers, communication infrastructure) can be reused from the control system (shared use), the integrated solution yields economic advantages. One further advantage of the integrated solution is that energy data, provided by actors, can be read and processed through the use of existing infrastructure. The drawback of such a solution is that a running automation system will be subject to changes in order to integrate the energy management functions.

The rest of this paper will focus on the integrated solution and it will describe an approach how an integrated solution can provide energy information for a tEnMS with minimized engineering effort.

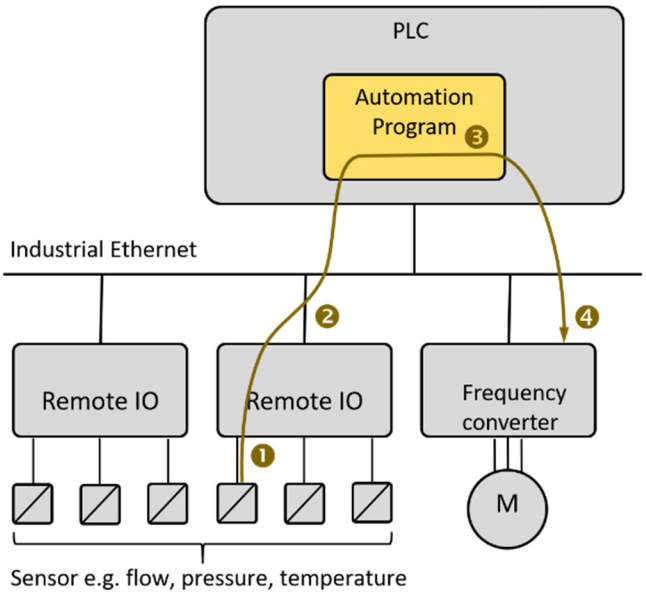

Fig. 1. Control Application

Fig. 1 shows a control application, that shall serve as example. A flow sensor measures the flow of a medium in a

DOI of the original publication: 10.1109/ETFA45728.2021.9613238

pipe ❶. The remote IO conveys the data via an Industrial Ethernet to the PLC ❷. The PLC calculates an output value ❸ and sends it to the frequency converter ❹ that controls the speed of a pump motor. Up to now energy data is not yet considered.

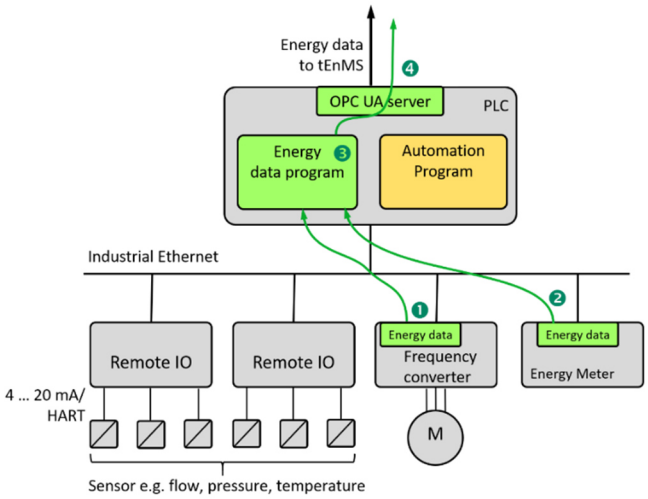

Fig. 2. Energy information processing

Fig. 2 now assumes that the frequency converter ❶ is able to provide energy data (e.g. current power, cumulated energy, etc.). In addition, it is assumed that an energy meter ❷ provides additional energy information. An energy data program in the PLC ❸ acquires the energy data, processes it and submits it to an OPC UA server ❹ on the PLC. This server provides the information for a superordinate energy data acquisition system that is part of the tEnMS. The energy data program reads the measured values from the sensor/ actuator level as raw values usually in a device specific format. The energy data program in the PLC then converts the raw values into a format, suitable for the energy management at the superordinate layer. To achieve this, the programmer needs to understand the data format that the devices provide, e. g. by reading the user manual of the device.

For some Industrial Ethernet protocols, energy profiles, such as PROFIenergy [4] and Sercos Energy [5], have been defined in order to provide a standardized format for energy data. This offers the advantage that no device specific documentation needs to be used to determine the semantics of the incoming energy data. However, only few devices support such an energy profile today. Based on the described situation, the following challenges arise:

- The energy data is provided from different devices from different manufacturers. These devices do not have a standardized semantic, unless energy profiles are supported by the devices.
- The energy data is provided via Industrial Ethernet or fieldbus. Depending on the industrial Ethernet / fieldbus protocol (e. g. PROFIBUS, PROFINET, Ethernet IP, etc.), the energy data gets into the controller in diverging data formats.
- The energy data program must consider the individual characteristics of the devices that provide the energy data. The programmer must retrieve this information from the respective device documentation and adapt the energy data program accordingly.
- The scaling of the raw values must be performed individually for each device.

Due to the aforementioned challenges, the implementation of a tEnMS involves considerable engineering effort to retrieve data format information and to write the energy data program for the PLC. In practice, this increased engineering effort leads to the situation that energy management functions are implemented only sporadically in industry [6]. These issues have been addressed in the research project *User-guided engineering of integrated energy management systems in production (EnEng_PRO)*[1]. The main objective is, to reduce the engineering effort required to implement energy data acquisition software. In the context of the project, a code generator is being developed that can be used to automatically generate specific energy data programs for PLCs. The code generator is intended to support all field devices of an automation system, also without energy profiles. The automatically generated PLC programs can then be imported and integrated into the corresponding engineering environment.

II. State of the Art (Related Work)

This chapter presents related work on the topics of automatic code generation of energy data programs, data exchange formats and interfaces.

A. PLC code import interfaces and application programming interfaces (API)

The program code of a PLC typically consists of a PLC program and/or a hardware configuration and other parts. Engineering tools offer interfaces that allow a manual import and/or export of program and configuration files. The interfaces on the one hand support proprietary data formats, which are tailored to the specific PLC engineering environment, e.g. [7]. On the other hand, there are standardized data interfaces, like PLCopen XML [8], in place that are vendor neutral. As an alternative to import and export of configuration files, application programming interfaces (API for short), such as described in [9], are provided. With the help of such an API, it is not only possible to import PLC programs and hardware configurations, but to place them directly at the desired location inside the project tree of the PLC engineering system, whereas with manual import the contents must be inserted by the user at the correct position in the project.

B. Integrated plant engineering to increase energy efficiency

In [10] [11] the development of code generators for the automatic creation of PLC programs is described, but this literature does not address the generation of energy management programs. These code generators support interfaces of the PLCopen XML format or proprietary formats [9]. Based on the possibility of performing an automatic code generation for PLC programs, [12] describes the development of a software tool that uses a code generator to automatically generate an energy PLC program that supports the PROFINET communication protocol and the PROFIenergy energy profile. The automatic code generation of an energy PLC program is designed to reduce the engineering effort for

[1] Funding code: ZF4283003RR8; Funded by: Federal Ministry for Economic Affairs and Energy based on a resolution of the German Bundestag

manual program generation. For the code generation, the code generator requires the engineering data and device description data of all field devices that are to be integrated into the tEnMS. As an input file for the code generator, a file in Automation Markup Language format [13] is provided, in which a network structure with one or more PROFIenergy nodes is contained. The entries for the PROFIenergy devices each contain a reference to the device-specific GSD files. Based on the information from the GSD files, it is automatically determined which PROFIenergy functionalities (e.g. measurement data acquisition or standby management commands) are supported by the device so that a suitable energy PLC program can be generated for the respective device of the network structure. The PLC program parts are generated in a generic structure, while PROFIenergy enables a semantically uniform mapping of the energy data. The energy PLC program created by the code generator is generated in an XML format, standardized by PLCopen [8]. The PLCopen format describes a manufacturer neutral exchange format that is compatible with PLC programs according to IEC 61131-3 [14]. Thus, the generated energy PLC program can be imported into engineering tools that support the PLCopen import. During operation, the developed energy PLC program communicates with higher levels by means of an OPC UA server running on the PLC to receive commands for load management and to provide the acquired energy data. These two known concepts shall be used to generate the energy data (acquisition) program for any kind of device, especially those, that do not support an energy management profile like PROFIenergy.

III. Design and Conception of the Code Generator Software

This chapter integrates the energy data program (EDP) generated by the code generator. The program structure of the EDP is then described. To this end, the process of creating the EDP, including importing it into the specific engineering environment, is first described in general terms, followed by a detailed explanation of the code generator software.

A. *Integration of the generated energy data program into the automation system*

The code generator software developed in this publication is responsible for the automatic generation of an EDP tailored to a specific automation system. The code generator is designed to reduce the engineering effort (see chapter II. B.) in the development of EDP. Due to the concept, the developer must create each device type specific EDP only once manually as a template file. Based on these template files, the code generator generates automatically an EDP for existing or new plants. The EDP automatically generated by the code generator software is to be integrated into the PLC in parallel with the automation program. Fig. 3 shows the data flow of the EDP data and the automation program data.

During runtime, the EDP and the automation software are to be executed in parallel in the PLC ❶ ❷. In contrast to the contribution presented in the chapter state of the art [12], the code generator software supports devices independent from specific energy profiles, such as PROFIenergy. Specifically, it is not necessary that the devices support any specific energy profiles. At least each device has to provide energy data. The EDP contains the energy related data of all devices of an automation system that provide energy data. The energy data of the frequency converter, for example, describes the current

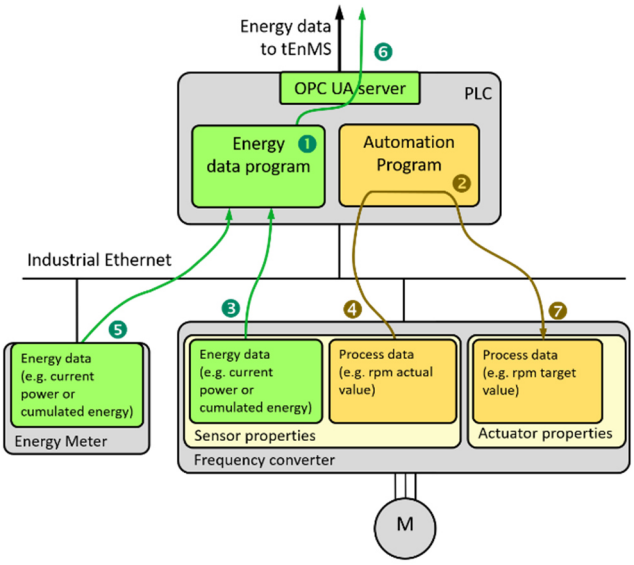

Fig. 3: Data flow of EDP data and automation program data

power flow ❸. All devices to be integrated into the EDP are assumed to provide energy data in addition to the process data ❹. A frequency converter provides process data such as the current number of revolutions of a motor per minute. Energy meters can be integrated as well ❺. In the PLC, the process data is linked to the automation program controlling the technical process and the energy program processes the energy data. Afterwards the energy program forwards the processed energy data to the OPC UA/ Modbus TCP server for further use by the tEnMS ❻ while the automation program sends determined commands to the frequency converter ❼.

B. *Energy data program structure*

This subchapter describes the program structure of the generated EDP. Fig. 4 shows the individual program elements.

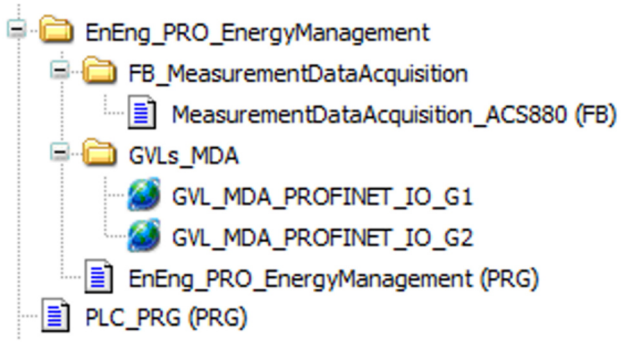

Fig. 4. Structure of generated PLC energy data program

The code generator creates and links all program elements during the EDP generation. IEC 61131-3 [14] defines most of the program elements as program organization units (POU). In this context, POUs are not only function blocks (FBs) but also programs (PRG). In addition to the POUs, the EDP requires global variables and a folder structure. The code generator instantiates a FB (e.g. MeasurementData-Acquisition_ACS880 in Fig. 4) for each device type that is part of the EDP. If there are several devices of one device type in the automation system, the code generator instantiates the corresponding FB multiple times. The instantiations take place in the EnEng_PRO_EnergyManagement (PRG) program. In order to provide the measured device energy data during runtime for higher level systems, the code generator

generates global variables (e.g. GVL_MDA_PROFINET-_IO_G1 and GVL_MDA_PROFINET_IO_G2) and connects them with the corresponding outputs of the instantiated FBs. Furthermore, the global variables for providing the measured energy data are linked to the OPC UA/ Modbus TCP server of the PLC. Also, the code generator links the inputs of the instantiated FBs to the corresponding hardware configuration. If the code generator software does not generate the hardware configuration, the already existing hardware configuration must be linked to the inputs of the instantiated FBs manually. All program elements generated by the code generator, apart from the hardware configuration, are integrated into a created folder structure (EnEng_PRO_EnergyManagement, FB_MeasurementDataAcquisition, GVLs_MDA) for better clarity and structuring. The engineering environment creates the main program (PLC_PRG), which is responsible for the automation of the industrial plant automatically. The user has to integrate the routine (EnEng_PRO_EnergyManagement (PRG)), which is responsible for the cyclic call of the EDP, into this main program manually.

C. *Overview of the configuration steps to be performed*

This subchapter generally describes the steps to be performed by the person using the code generator software. The code generator software generates the EDP on the basis of a project file. The project file contains the instantiated devices that support the provision of energy data. Each instantiated device describes a component located in the real plant (e.g. a frequency converter). The user manually instantiates each device in the project file. To do this, the user selects the specific device type to be instantiated from the library integrated in the code generator software. The library contains all different types of devices that the user has ever added to it. If a device type is not yet integrated, it can be added via an editor. The user then configures the selected device so that it corresponds to the real device of the plant. After all devices have been successfully added, the user starts the code generation function. Based on all devices instantiated in the project file and the IEC 61131-3 standard [14], an EDP is generated in the structured text programming language automatically. The code generator stores the program in XML-based PLCopen format [8]. The following Chapter “Structure and operating principle of the code generator” describes the structure of the code generator in detail. In addition to the EDP generation, there is an optional possibility that the code generator performs the target system specific hardware generation of all instantiated devices. The optional hardware generation is useful, if the user wants to integrate the EDP into a PLC of a new plant where no hardware configuration is available yet. The code generator stores the hardware description generated also in the XML-based PLCopen format. Following the generation, the user imports the EDP file and, if necessary, the hardware configuration file into an engineering environment that has an import and export file interface of the PLCopen standard [8]. If the code is generated for an already configured system, the hardware configuration and automation program are already available in the engineering environment. In this case, the user must only integrate a method for the execution of the imported EDP into the main program of the PLC. Finally, the user downloads the automation program, the EDP and the hardware configuration from the engineering environment into the PLC. Fig. 5 shows the described procedure.

After successful implementation of code and hardware generation for engineering environments the development of

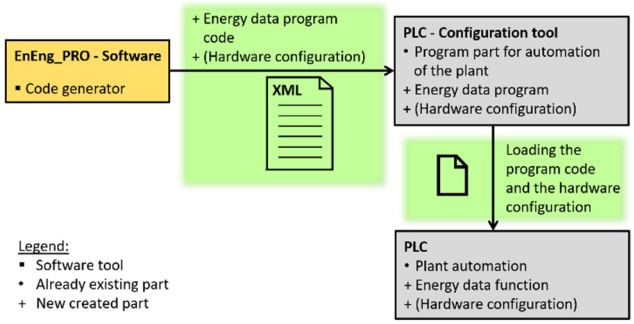

Fig. 5: Code generation overview

code and hardware generation for the specific application programming interface of proprietary engineering environments follow. Proprietary engineering environments are characterized by the use of proprietary import and export standards. For example, for a particular proprietary engineering environment the code generator software must store the EDP code in a proprietary XML file format [9]. If the EDP code is not stored in a proprietary XML file format, an import into the specific engineering environment is not possible. The same proprietary engineering environment uses Computer-aided x (CAx) files of the Automation Markup Language (AML) type as the import file format for the hardware configuration [9].

D. *Sructure and operating principle of the code generator*

To generate the described energy data program automatically, a code generator is needed. This subchapter describes the structure and operating principle of the EnEng_PRO code generator software in detail. To understand the operating principle of the software, the terms device object, device object type, device object type library, device object type library editor and device instance list, need to be explained first. Fig. 6 shows the schematic of the code generator.

A device object ❶ represents a device located in the real automation system, e.g. a frequency converter of a certain type, which is to be integrated into the EDP. Each device object consists of two parts. The first part contains unchangeable properties. These unchangeable properties are described in the device object type ❷. The second part contains individually configurable properties, which are called dynamic properties. These dynamic properties are described in the device object itself ❸. The device object type properties are always identical for each device object of the specific type. These unchangeable properties are specific type information such as the name of the model and the manufacturer, as well as the measured variables supported by the model (e.g. current, voltage, current ambient temperature, etc.). Other unchangeable properties are the supported communication protocols. So far, the code generator software supports devices that use the communication protocols PROFINET IO and/or PROFIBUS DP. In addition, template files describe each device object type. On the one hand there is a specific template file for each communication protocol type, which is necessary to generate the hardware configuration. On the other hand, there is a template file, which contains the device specific energy management function. This template file is necessary to generate the EDP. The code generator software stores all device object types in the device object type library ❹ so that they are available for reuse. Editing of all device object type entries in the device

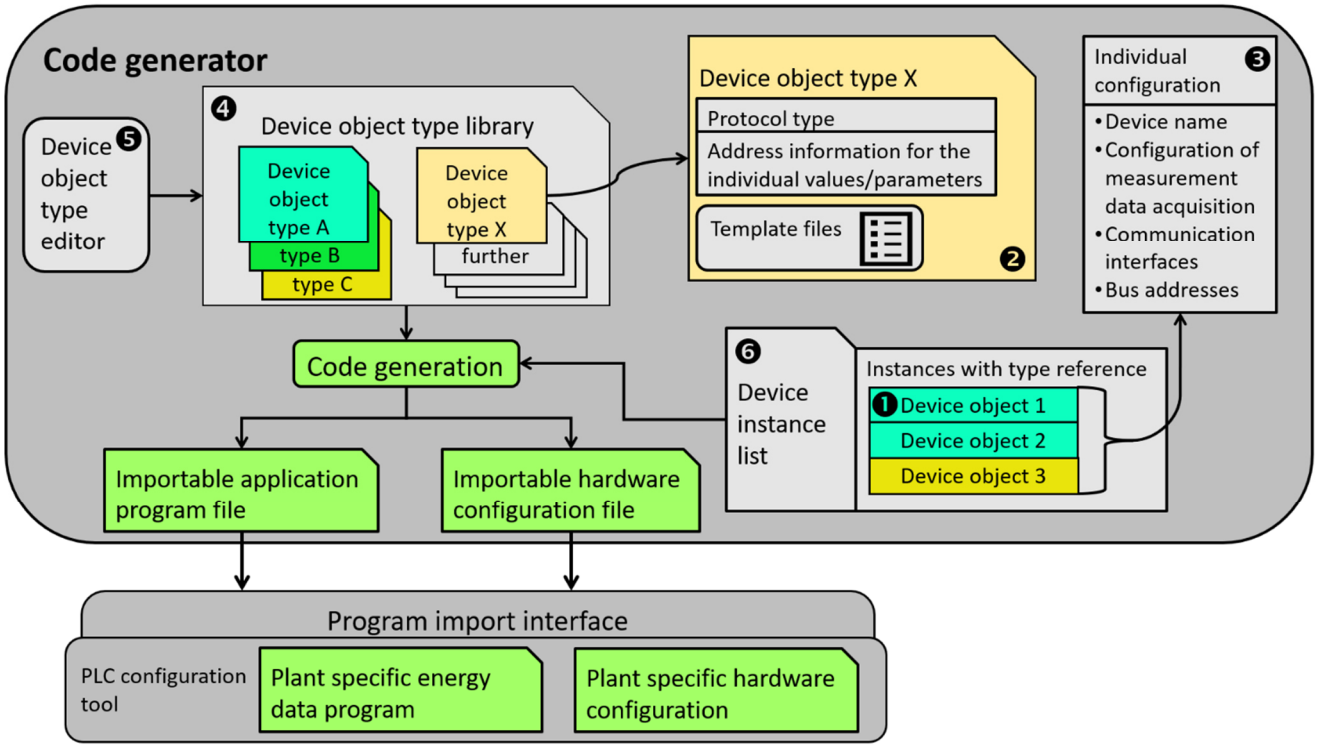

Fig. 6: Used data model in the code generator

object type library is possible by using the device object type library editor ❺. Specifically, new device object types can be added to the device object type library, existing entries modified or obsolete entries removed from it. When the user is adding a new device object type entry to the library, the user has to import the device specific template files during this process. First the user must create each template file once in the engineering environment to be supported. After that, the user exports the file in the engineering environment specific file format (e.g. PLCopen XML). After successful completion of the device object type creation process the code generator software stores the template files in a predefined folder.

Dynamic properties are the individual device name, the selection of the communication protocol used by the device in the plant and the unique protocol address. Another dynamic property is the selection of measured variables. The measured variables supported by the device object type, which should be included in the EDP are selected here.

All device objects created by the person using the EnEng_PRO software, including their respective dynamic components, are instantiated in the device instance list ❻. The code generator connects each instantiated device object with the associated device object type. During the code generation process, the code generator generates the target system specific EDP on the basis of the instantiated device objects, as well as the reference to the respective device object types. For this purpose, the code generator fills all required template files with the configuration parameters entered by the user. The filled template files are then used to generate the EDP for PLCopen based systems or systems that uses a specific application programming interface. Optionally, the code generator generates a target system specific hardware configuration as well, which is also created on the basis of filled template files.

IV. Design of the User Interface

The code generation of the EDP requires a lot of configured device objects. To configure each device object intuitively, the user needs a user interface that supports the easy use of the code generator software. The code generator software stores each configured device object in the device instance list of the actual EnEng_PRO project.

Fig. 7 shows the screen with a device object instance list containing three device objects ❶ ❷ ❸.

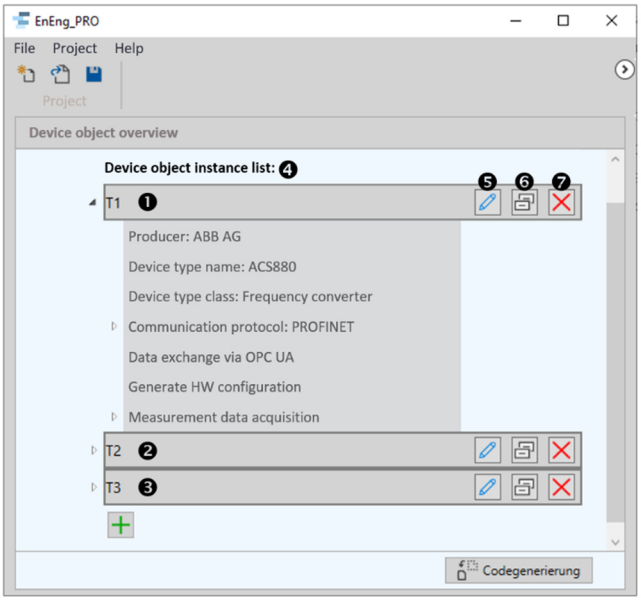

Fig. 7. EnEng_PRO user interface with device object instance list

In the user interface the user can add device objects to the device object instance list ❹ or edit ❺, duplicate ❻ or delete

❼ existing device objects. Expanding the tree view node for device object T1 ❶ results in displaying related information and settings for the device object. The device object type ACS880 was selected from the device object type library. For this device object type, some general information such as manufacturer or the device type class (in the example frequency converter) is taken from the library. In addition, the library entry also contains information about which measured variables can be acquired by the corresponding device as well as the supported communication interfaces. The device type data from the library is linked to the PLC source code template and the individual configurations in the user interface. The code generator software generates PLC EDP which contains each created device object. Through the user interface the following features are supported:

- EnEng_PRO project management (e.g. to save or open a configured device instance list as an EnEng_PRO project file)
- Selection of a device object type from the library to retrieve device specific data and the associated PLC source code template
- Extending and customizing the library with the device object type editor
- Optional generation of the PLC hardware configuration
- Selection of the communication interface of the device to be used (including bus specific address settings, which are included into the hardware configuration)
- Selection of the communication interface to provide the energy data to higher level systems on the PLC (e.g. OPC UA, Modbus TCP)
- Selection of energy data elements to be used as basis for code generation
- Selection of a PLC code import format (e.g. PLCopen XML)
- Automatic generation of a PLC energy data program for energy data acquisition

V. Conclusion

This publication describes the development of the EnEng_PRO code generator software as a part of the tEnMS. This software generates the plant specific energy data program (EDP) for each PLC automatically. The EDP acquires the current energy data of each device supporting energy functions connected to the PLC. Then the EDP processes the energy data. The provision of the processed energy data to higher level systems takes place via OPC UA/ Modbus TCP server of the PLC. The code generator supports PLCopen based PLC systems [15] as well as systems that provide a system specific application programming interface [9]. The code generator uses the existing hardware structure information of the plant, in order to integrate all devices that are providing energy data in addition to process data (e.g. frequency converters) into the EDP. Energy meters can be integrated as well. The code generator software has a user interface designed to support the automatic creation of an EDP. In the first step, the EDP is responsible for the energy data acquisition of all devices of an automation system that support the provision of energy data, independent of energy profiles. These devices must support the communication protocols PROFINET IO and/or PROFIBUS DP. The EnEng_PRO software stores all different types of device description information in a permanent library. The user uses the library entries to configure the energy data program. An editor allows modifying any library entry, as well as adding new entries or deleting obsolete entries. With a growing number of energy management projects carried out, this leads to a steadily growing number of library entries. The number of library entries is growing because each automation system has implemented different device object types. Some of them are not yet implemented in the library, but are needed for the EDP. A constantly growing number of library entries leads to a continuously increased reuse of the individual device object types. That leads to a continuous reduction of the engineering expenditure. It is assumed that a low engineering effort increases the willingness to integrate a tEnMS into the automation system. A successful integration of a tEnMS is assumed to reduce the energy consumption of the plant and thus energy cost. In addition, a contribution is made to climate protection.

In the next development steps, the code generator can be extended by additional functions. For example, it is possible to add load management functions. Load management functions can switch devices of an automation system into different energy saving modes.

References

[1] German Federal Association of Energy and Water Industries: Electricity price for industry. Aug. 2019. At: https://www.bdew.de/service/daten-und-grafiken/strompreis-fuer-die-industrie/ (20.03.2021).

[2] ISO 50001: Energy management systems - Requirements with guidance for use. 2018.

[3] Rossiter, A. P., Jones, B.P. (Eds.).: Energy management and efficiency for the process industries, Wiley American Institute of Chemical Engineers, Hoboken, 2015.

[4] PROFIBUS Nutzerorganisation e. V. (PNO): Common Application Profile PROFIenergy: Technical Specification for PROFINET. Version V1.3. Oct. 2019. At: https://www.profibus.com/download/profienergy/ (19.03.2021).

[5] Schlechtendahl, J.: Whitepaper SERCOS Energy. Feb. 2012. At: https://dc-us.resource.bosch.com/media/en/general_use/products/engineering/sercosiii/files_1/sercos_energy_whitepaper_en.pdf (24.03.2021).

[6] Güldner, F., Menze, T. ARC Advisory Group - Energy Management Online Survey. The Findings, ARC Advisory Group, 2014.

[7] Rockwell Automation: Logix 5000 Controllers Import/Export. Sep. 2020. At: https://literature.rockwellautomation.com/idc/groups/literature/documents/rm/1756-rm084_-en-p.pdf (22.03.2021).

[8] PLCopen: Technical Paper PLCopen Technical Committee 6. XML Formats for IEC 61131-3. Version 2.01 - Official Release, 2009.

[9] Siemens AG: Openness Automating creation of projects. Oct. 2018. At: https://cache.industry.siemens.com/dl/files/163/109477163/att_926042/v1/TIAPortalOpennessenUS_en-US.pdf (24.03.2021).

[10] Steinegger, M., Zoitl, A.: Automated code generation for programmable logic controllers based on knowledge acquisition from engineering artifacts: Concept and Case Study. In: Proceedings of the 17th IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA). Poland, Karków, 17. – 21.09.2012.

[11] Armentia, A., Estevez, E., Orive, D., Marcos, M.: A Tool Suite for Automatic Generation of Modular Machine Automation Projects. In: Proceedings of the 16th IEEE Int. Conf. on Industrial Informatics (INDIN). Protugal, Porto, 18. – 20.07.20189. P. 553-558.

[12] Würger, A., Niemann, K.-H., Fay, A., Gienke, M., Paulick, M.: Integrated plant engineering to increase energy efficiency. In: atp Nov.- Dec. 2019. P. 70–77.

[13] Lüder, A., Schmidt, N.: AutomationML in a Nutshell. Nov. 2015.At: https://www.automationml.org (22.03.2021).

[14] IEC 61131-3: Programmable controllers - Part 3: Programming languages. 2013.

[15] ABB AG: Automation Builder. At: https://new.abb.com/plc/automationbuilder (30.03.2021).