Extracting Energy Data from MODBUS Devices Using CIP

Rick Blair
System Architect
Schneider Electric

Presented at the
2012 ODVA Industry Conference & 15th Annual Meeting
October 16-18, 2012
Stone Mountain, Georgia, USA

Abstract

As increased importance is placed on measuring and managing energy, companies are making energy management a key component of their control systems. In response, the ODVA Energy Special Interest Group (SIG) has been working over the past two years on defining a standard way to represent energy within the Common Industrial Protocol (CIP) model. As a result of this work, several new CIP objects have been defined to offer energy information in a common form.

While these new energy objects will standardize how energy is represented in CIP devices, the current marketplace is filled with MODBUS based energy devices from many vendors. Unlike CIP, which is object-based, MODBUS is based on a flat memory structure. No standard method of representing energy is present, either within or across manufacturers. This requires consumers to learn the MODBUS layout of each of these devices or to use proprietary software from the device manufacturer to extract relevant energy data from their energy devices.

This paper will explore the definition of a device that would extract MODBUS energy data from various energy devices and place it into common CIP energy objects. Ideally, this would be best accomplished for legacy MODBUS products without any modification to the device, but it will be shown why this is not always possible. Use of Electronic Data Sheet (EDS) or Extensible Markup Language (XML) files to define the MODBUS device is also considered. Also considered are modifications that could be made to MODBUS energy devices that would enable direct interrogation to determine its data formats and mappings.

In addition to defining where the data is located within a MODBUS energy device, there is the challenge that data may not be in the same format. Data conversion between the various formats will also be discussed.

Energy devices based on the MODBUS protocol will continue to be developed and sold, regardless of the improvements made in the CIP protocol. Hopefully, the ideas and recommendations presented in this paper will allow the creation of a data extraction device that can be used by those customers that want to manage and optimize their energy usage using CIP.

Keywords

MODBUS
CIP
Energy

During the last two years, three standard energy objects have been defined in the CIP specifications (Base Energy Object, Electrical Energy Object and Non-Electrical Energy Object). With these standard ways to represent energy, tools will no longer need to rely on special configurations to know where energy data is stored as is the case with today's MODBUS devices. The following paragraphs summarize these objects.

Base Energy Object – Class Code: 4E Hex

The Base Energy Object acts as an "energy supervisor" for CIP energy implementations. It provides a time base for energy values, provides energy mode services, and can provide services for aggregating energy values up through the various levels of an industrial facility. It also provides a standard format for reporting energy metering results. The object is energy type independent and allows energy type specific data and functionality to be integrated into an energy system in a standard way. Multiple instances of the Base Energy Object may exist in a device. For instance, an electric power monitor may count the metering pulse output transitions of a separate metering device. The count of such transitions, represented by a Base Energy Object instance, would reflect the energy consumption measured by the separate device. As another example, a device may act as a proxy for the energy consumed by a number of simple devices, where each device is associated with a separate Base Energy Object instance.
An instance of the Base Energy Object may exist as a stand-alone instance, or it may exist in conjunction with an Electrical and/or Non-Electrical Energy Object instance. If an instance of the Electrical and/or Non-Electrical Energy object is implemented in a device, it must be associated with a Base Energy Object instance in the device.

Electrical Energy Object – Class Code: 4F Hex

This object is used to provide unified electrical energy management capability for CIP-enabled devices and processes. Energy management is typically related to the measurement and reporting of a variety of metering results. This object provides for the consistent reporting of electrical energy data. Electrical energy is assigned a separate object to accommodate its alternating and polyphase characteristics, which results in a collection of attributes that are unique among energy sources.
Instances of the Electrical Energy Object must be associated with an instance of the Base Energy Object.

Non-Electrical Energy Object – Class Code: 50 Hex

This object is used to provide unified non-electrical energy management capability for CIP-enabled devices and processes. Energy management is typically related to the measurement and reporting of a variety of metering results. This object provides consistent reporting of non-electrical energy data, including, without limitation, natural gas, fuel oil, steam, compressed air, hot water, chilled water, etc.
Instances of the Non-Electrical Energy Object must be associated with an instance of the Base Energy Object.

MODBUS to CIP Energy Data Extractor Device

The remainder of this paper will focus on the functions required in a MODBUS to CIP Energy Data Extractor (MCEDE) device including exploration of several implementation alternatives. Design details are not described, but rather a high-level overview is presented.

Figure 1 shows a basic overview of an MCEDE. The functionality could be implemented in a dedicated platform or, with appropriate interface cards, an application running on a standard PC. What is needed is an interface to MODBUS energy devices, which could be either RS-485 serial or Ethernet. For the CIP interface, it could be any CIP network (e.g. EtherNet/IP, DeviceNet, etc.). When both MODBUS/TCP and EtherNet/IP are present, a single Ethernet interface could be used, but it may be better to have separate interfaces in order to isolate network traffic. Finally, for a dedicated platform, there needs to be an interface to allow a user to configure the MCEDE and to add MODBUS energy device description files. If an Ethernet port is present, this could be done via Web pages.

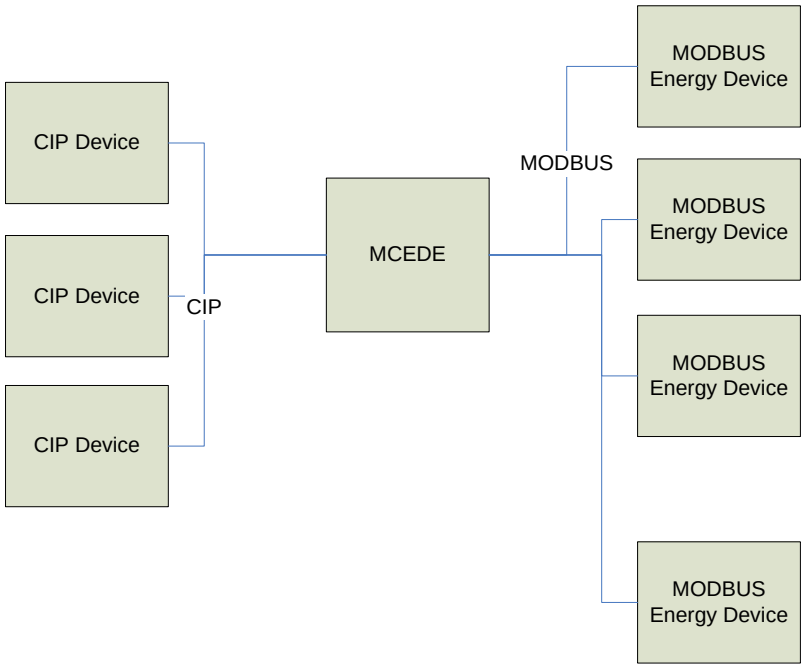

Figure 1 – MCEDE Overview

Basic Functionality of an MCEDE

The basic building blocks of an MCEDE are as follows:

- A set of MODBUS energy device descriptions
- A set of data type conversion functions
- A configuration function
- A method to add/delete MODBUS energy device descriptions
- A scan function to search for MODBUS devices (optional)
- A MODBUS interface
- A CIP interface
- An energy object service handler

Describing the MODBUS Energy Device

Unlike CIP, which is object-based, MODBUS is a flat architecture with data represented as 16-bit words (registers) or bits. In fact, the MODBUS protocol does not even define how to handle data larger than 16 bits. This definition is left to the application present within the MODBUS device. Designers of products based on the MODBUS protocol have free reign on the locations and order of multi-register data within this flat architecture. As a consequence, the many MODBUS energy devices currently in the field have no common organization for the energy data within this register space. This is true both across and within various company offers. Because of this, it is necessary to describe where and in what format the MODBUS energy data exists for any MODBUS energy device connected to the MCEDE.

In order to effectively extract energy data from a MODBUS energy device, it is necessary to describe several aspects of the MODBUS energy device. The MCEDE can then use this description to collect the MODBUS energy data and present it in the appropriate CIP energy objects. The following should be considered when describing a MODBUS energy device:

- Which MODBUS data should be extracted
- Number and type of CIP energy objects needed
- Order of multi-register data values

- MODBUS register blocks to read
- How to identify the MODBUS energy device
- Which CIP energy object services should be provided

And for each data element:

- MODBUS register address
- Data type
- Units

Two methods are considered for describing a MODBUS energy device. The first works with existing devices without modification. Like CIP devices, a file can be created to describe the MODBUS energy device, including how to identify the MODBUS device, location of energy data, its data types and its mapping to CIP energy objects. There are two logical choices, either EDS or XML files. A current challenge with using EDS files is that it requires an ODVA vendor number and for the vendor to assign a product code for each MODBUS device. A vendor must license ODVA technology in order to obtain a vendor number, which may not be cost effective for some MODBUS vendors. Additionally, a method to describe transformation of a MODBUS value into a corresponding CIP value would need to be defined in the EDS format. Another approach would be to use an XML file to describe the various attributes of a MODBUS energy device. One possibility is to base the XML file format on the device profile model defined in the International Standard ISO 15745-1.

A second approach involves defining a method for a MODBUS energy device to describe itself via MODBUS commands. This requires modification to the MODBUS device and is considered later in section **Embedded MODBUS Device Profiles**. This paper will focus on MODBUS device descriptions that exist in external files.

To accommodate MODBUS energy devices produced in the future, the MCEDE should have the ability to import MODBUS energy device descriptions.

A single MODBUS device may be represented by a single set of CIP Energy Objects or multiple sets of Energy Objects (See Figure 2). The MODBUS energy device description should specify the quantity of CIP Energy Object instances needed to represent the MODBUS energy device. In addition, some CIP energy object attributes may not exist in the MODBUS energy device. For required CIP energy object attributes that are not present in the MODBUS energy device, the MODBUS device description needs to specify how the MCEDE should provide that required attribute. For optional attributes that are not present in the MODBUS energy device, the MODBUS device description should indicate which optional attributes need to be provided by the MCEDE.

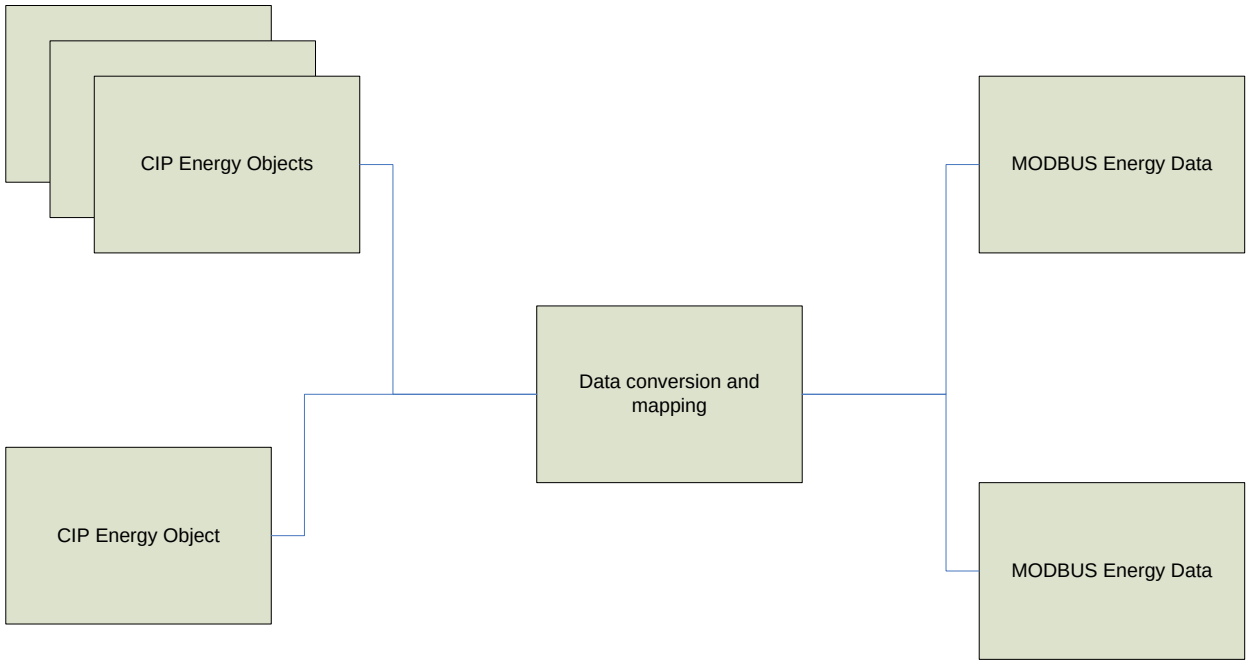

Figure 2 – Mapping Concept

The order in which multi-register values are represented in a MODBUS energy device must also be described in the MODBUS energy device description. The MODBUS protocol is only based on bit and register values and does not consider values larger than 16-bits. The format of the data is specified across the wire for single MODBUS registers as most significant byte first. This only helps to reconstruct 16-bit register values. If data items are greater than one register, like 32-bit or 64-bit floating point values, long integers, etc., there is no defined standard for how these values are represented. The application within the MODBUS energy device can choose how multi-register data is represented. It is, however, assumed, that the representation is consistent with a MODBUS application, so that the register order for multi-register data can be described only once in the MODBUS device description and need not be indicated for each data item.

Describing the MODBUS Device Data Structure

Creating the entries in the MODBUS energy device description that describes the data structures of a MODBUS energy device and how it maps to CIP Energy Objects is a one-time manual process. The first step is to determine the type (base, electrical and non-electrical) and quantity of CIP Energy Objects that are needed to represent the MODBUS energy device. At least one Base Energy Object is needed for each MODBUS energy device connected to the MCEDE. For the case when none or few of the Base Energy Object attributes are available in the MODBUS energy device, the MCEDE needs to calculate those attributes. In this case, the MCEDE would be considered a proxy for the base Energy object data, as described in the Base Energy Object definition.

Optimizing the MODBUS Communication

After the mapping of MODBUS energy data to CIP Energy Objects is determined, the MCEDE could simply perform single MODBUS requests to each data item. However, a more efficient approach would be to minimize the number of MODBUS requests to get the data as suggested in Figure 3. This involves analyzing the grouping of the necessary data so each MODBUS request gets multiple data values. In fact, it may be more efficient to even read data that is not used in the CIP objects in order to minimize the number of MODBUS request. For simple MCEDE devices, this could be done manually as part of the mapping process. A more advanced MCEDE could have built-in algorithms to analyze the MODBUS data organization for the most efficient data retrieval method.

Regardless of the approach, it is unlikely that data within these blocks of registers is in the same order as that of the CIP objects, so it will be necessary to describe at what offset in these blocks the various attributes are located.

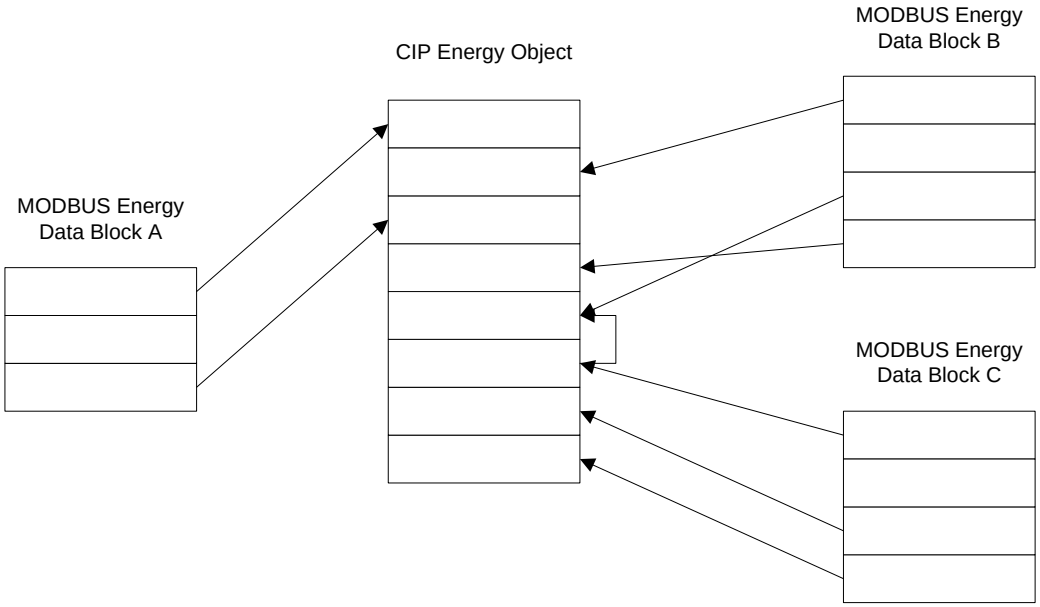

Figure 3 – MODBUS Mapping to CIP Object

Once the location of each MODBUS data item is known, it is necessary to describe additional properties of the MODBUS data so it can be converted, if necessary, to the format expected by the CIP energy objects.

For each MODBUS data item, the following information is needed:

- MODBUS register address
- Data type
- Units

Armed with the above information, the units and data types of the MODBUS registers can be compared with the units of the CIP object attributes. If they are the same, then no data conversion is needed.

Data Conversion

When the data from a MODBUS device is not in the same format as it is represented in the CIP object, it is necessary to describe how to convert the data so that the MCEDE can perform the conversion (e.g., integer to long integer, 64-bit integer to odometer, etc.). Within the MCEDE, conversion routines will then need to be implemented. Care should be taken in the design of these conversion routines, since data loss may occur. For example, assume that the MODBUS data for an odometer data type contains fewer digits than the CIP object attribute. In this case, the rollover in the MODBUS value will occur before it would in the CIP object. Should the MCEDE simply return to zero like the MODBUS value or should it continue to count? If the MODBUS value is internally represented as a binary value, the rollover may occur at a different point than would happen with the odometer data type. These are the types of situations that need to be considered when designing MCEDE data conversion algorithms.

Once the data type conversion has been chosen, additional conversion may be necessary if the units do not align. Because it would be impossible to predefine all unit conversion algorithms within an MCEDE device, simply provide a multiplication factor in the MODBUS device description file. This factor would indicate what the MODBUS value should be multiplied by to get the CIP object attribute value.

Describing how to Identify the MODBUS Device

Like CIP, there is a MODBUS message to uniquely identify a MODBUS device. Unfortunately, this method was introduced well after the establishment of the MODBUS protocol, so it is not present in many MODBUS

implementations. Some MODBUS devices use certain registers with specific values to allow tools to identify their device. Finally, some devices use no identification methods, so it is impossible to identify those devices using only the MODBUS protocol. The MODBUS device description file needs to take these various possibilities into consideration.

MCEDE design Considerations

The following sections describe in more detail design considerations for the MCEDE.

Configuration

In order to use an MCEDE device, it must be configured. There are several aspects to consider. First is to configure the communication parameters (e.g. IP address for EtherNet/IP and/or MODBUS TCP/IP, baud rate for MODBUS serial communication, etc.). Management of MODBUS energy device configuration files is also needed. It must be possible to add and remove MODBUS configuration files as needed to support the connected MODBUS devices. Finally, functionality needs to be provided that allows the user to indicate what MODBUS devices are connected to the MCEDE, at what communication address, how frequently it should be scanned, to what MODBUS device the description applies, and to what CIP energy object instances the data should be mapped.

Identifying the MODBUS Device

It may be desired that the MCEDE can automatically determine what MODBUS devices are connected to it or, at the very least, make sure that the MODBUS energy device matches the description file. Newer MODBUS devices support a consistent method for device identification, as described in the following section.

The MODBUS protocol defines a Read Device Identification (RDI) function. The Basic RDI function code allows reading the identification (VendorName, Product code, and revision number) of a MODBUS device.

It is possible that a MODBUS device does not support the RDI function. In this case, the device description file should provide an alternate method to identify the device, such as reading specific MODBUS registers and looking for specific return values. Reading of these registers can then be used to determine the identity of the connected MODBUS device.

Finally, a MODBUS device may have no method to determine its type. In this case, the MCEDE cannot find the device during a scan nor verify that it matches the device description file. Fortunately, devices like this are becoming rarer in the marketplace.

The fact that there are several methods that can be used to determine the ID of a MODBUS device makes automatic scanning for devices complicated. For the range of addresses to be scanned, the MCEDE will need to try all possible identification methods resident in its repository of device description files. This could be time-consuming, so letting a user choose which algorithms to use for automatic scanning may be a worthwhile enhancement.

Updating CIP Object Data

The MODBUS protocol does not have a feature that is similar to CIP's implicit messaging. It only supports messaging similar to CIP explicit messaging. Hence, the MCEDE will need to poll each MODBUS device to get its energy data. The design of a method to poll the devices can be as simple as get all the data from the first device, move on to the next, etc. until all devices are read, then start over. A more complicated approach could be to allow the user to set an update time rate for each device and then design a scan algorithm that could handle this flexibility. As each data block is read, any necessary multi-register item register swapping should be applied prior to any data conversion and mapping activities. Care should be taken to ensure that all writes to CIP object attributes are autonomous, i.e. no access permitted to the attribute during its update and no update during access. If memory size is not an issue, one approach could be to have two copies of each CIP object, and when an object is updated, the context is switched to the updated object and the older copy becomes the next update target. This could minimize the potential for mutual exclusion issues.

CIP Energy Object Services

There are several services common to all CIP energy objects (Get_Attributes_All, Get_Attribute_List and Get_Attribute_Single). The Get_Attribute_List service should be implemented in the MCEDE to allow a CIP client device to determine what attributes are supported, since this may not be a consistent list for each connected MODBUS energy device. Additionally, the Get_Attribute_Single and Get_Attributes_All should both be implemented to allow master devices the most flexible methods for accessing the energy data.

The Base Energy Object defines additional services. The MCEDE will need to manage a varied list of CIP energy objects based on what MODBUS devices are connected. As a consequence, the MCEDE should implement the following services.

- Set_Attribute_List
- Create
- Delete
- Set_Attribute_Single
- Get_Member
- Set_Member
- Insert_Member
- Remove_Member

In addition to the above services, the Base Energy Object also defines a Reset service, which can be used to reset various aspects of the Base Energy Object, such as Odometers and the object itself. The use of this service is discouraged in the CIP Base Energy Object definition, so it will also be discouraged for an MCEDE.

The Base Energy Object also defines object specific services for starting and stopping metering. The MCEDE should implement these services, since it may act as a proxy and provide the metering for MODBUS power monitoring devices.

Embedded MODBUS Device Profiles

As described earlier, one approach to defining a MODBUS energy device is with a MODBUS energy device description file. This approach has an advantage in that no modifications are needed to the MODBUS energy device. However, for new MODBUS devices or those undergoing modification, an alternate approach could be to embed the same information into the MODBUS device. This would allow an MCEDE to get the necessary information directly from the MODBUS device rather than rely on a configuration file. Three approaches are described.

Embed MODBUS Device Description File in MODBUS Device

The MODBUS protocol provides a function code to read files. Once a MODBUS energy description file is created, it could be embedded in the MODBUS energy device and then the MCEDE could simply read the file directly from the device, rather than relying on the user to import the file into the MCEDE.

Use Predefined Signature and MODBUS Registers to Describe Data

Another method could be to describe, using MODBUS registers, the locations and data types of the various energy data items that an MCEDE could extract. These would need to be predefined and the MODBUS energy device could have a special signature, again using predefined MODBUS registers, to indicate its usage of this technique. An example of such a method is the SunSpec Alliance Interoperability Specifications, which describe data information models and MODBUS register mappings for devices used in Renewable Energy systems (refer to www.sunspec.org).

Extend MODBUS Function Code

The MODBUS protocol defines a Read Device Identification (RDI) function, which is the only function in MODBUS that is object-based (i.e., the use of this function does not require any information on the target device's memory model). It will be used as the basis for the following approach.

The RDI function code allows reading the identification and additional information relative to the physical and functional description of a MODBUS device. The Read Device Identification interface is modeled as an address space composed of a set of addressable data elements. The data elements are called objects and are identified by an object ID.

The interface consists of 3 categories of objects:

- Basic Device Identification. All objects of this category are mandatory: VendorName, Product code, and revision number.
- Regular Device Identification. In addition to Basic data objects, the device provides additional and optional identification and description data objects. All of the objects of this category are defined in the MODBUS Application Protocol Specification but their implementation is optional.
- Extended Device Identification. In addition to regular data objects, the device provides additional and optional identification and description private data about the physical device itself. All of these data are device dependent.

The focus of this description relies on using the Extended Device Identification capability of the RDI function. Similar to the MODBUS energy device description file, the various objects in the Extended Device Identification area could be predefined to contain the necessary information to allow the MCEDE to extract the salient MODBUS energy information. It would be impossible to predefine each object, since there is a wide range of MODBUS energy devices with varying capabilities. Hence, only a small subset of Extended Device Identification objects should be predefined (such as those that are global for the MODBUS energy device) and then these objects themselves can contain links to the remaining data elements. The MCEDE could first interrogate the MODBUS energy device to see if it supports this method, and, upon success, interrogate further to build its description of the MODBUS energy device.

Conclusion

This paper presented an overview of a MODBUS to CIP Energy Data Extractor. While not going into specific design and implementation details, it presented the features and functions needed in an MCEDE. Hopefully, the information presented can be used as a starting point for the creation of such a device.

References (optional):

- ISO/DIS 15745-1, Industrial automation systems and integration – Open systems application integration framework – Part 1: Generic reference description
- MODBUS Application Protocol Specification – V1.1b
- SunSpec Alliance Interoperability Specification – Common Elements Version 1.5 Draft 8