

# Object oriented P-NET

Objects in Delphi, VIGO  
and P-NET

By Ole Cramer Nielsen.

## Introduction:

- The interpretation of terms such as "Object oriented" and "Object oriented programming" can vary between programmers who use object oriented programming languages. I will therefore attempt to define my understanding of object oriented.

## Object definition.

- The important definitions in relation to this paper are as follow:
- An object can be declared with a number of “properties” and “methods”.
- An object with its “properties” and “methods” holds program routines and data.
- Data held within the object can only be accessed through “properties” and “methods”.

## Object definition.

- “Get property” is used to read the value of a property.
- “Set property” is used to write a value into a property. This value can later be read with “get property”.
- “Method” is like a function known from Pascal, and other languages, with possible parameters and a return value.
- It is **not the task** of the accessing program to select the correct get/set routines.
- This is dealt with by the object.

## Class and instances

- Object “Class”, which is a type specification of an object.
- An object will first exist when an object instance of a specified class is created.
- Independent objects of a common class can be created as instances, each with its own data. If the objects are in the same device, the set/get routines are normally shared.
- Object instances can be created statically or dynamically.

## The procedural way

- This concept is in contrast to the procedural way, where the programmer has to study the different available procedures and select the appropriate one together with the correct data.
- The procedure for a given activity may vary from device to device. There is no protection against using the wrong procedure with the wrong data.

## Object oriented subjects

- “Object oriented” concept ” can be divided into four subjects:
  - Access of objects and their properties and methods.
  - Define static objects.
  - Dynamic creation of object instances.
  - Designing new classes.

## Access of objects

- The first step in object oriented programming is accessing properties and methods of objects within one program instance.
- Object Linking and Embedding, known as OLE and devised by Microsoft, enables access to properties and methods from within different program instances.
- DCOM enables access to properties and methods from different computers.
- This paper will discuss the remote "access of objects" via fieldbus.

## Procedural based fieldbuses:

- For many fieldbuses, including Profibus and World FIP, the basic principle on which the buses were built, originated from the procedural based MMS.
- A number of communication procedures are available in the remote device, identified by an index. (May vary from manufacturer to manufacturer).
- These procedures can be activated via the fieldbus, together with parameters to identify the data.
- It is the task of application program in the master/client, to combine the required procedure index with the correct data.

## Object oriented based fieldbuses

- P-NET uses the object oriented principle, where the communication procedures are reduced to "get" and "set" (Load and Store).
- In the server (slave) the SWno is used to identify the "set" routine, the "get" routine and data.
- It is **not** the task of application program at the master/server to have knowledge of get and set procedure indexes or their functionalities.
- The object oriented principle, makes it far simpler to create an application program, and also makes it easier to reuse them.

## Access to objects

- Here we discuss internal objects and remote objects, accessed from the programming language Delphi (based on Pascal), and describe what happens in VIGO, P-NET and in the slave.

## Get property – Delphi

- In Delphi, a Get routine for an internal object is activated as shown:
- *X := Object.property;*
- This “Get property”, sets “X” equal to a value held by the specified property of the specified object.

## Get property - VIGO/P-NET

- The PhysId to selects a remote property in an object
- When setting PhysId, VIGO converts the object and property identifier into a SWno using the MIB.
- VIGO supports "Get property" in the same way as Delphi:
  - $X := Object.value;$
  - "Get property", equates to "Assign X with the value of the selected property".

## Get property - VIGO/P-NET

- A “Get property” in VIGO, initiates a load instruction via P-NET, using a SWno to identify the property.
- The slave receives the request, converts the received SWno into a “Get routine” and a pointer to the data.
- The routine is executed, which returns the value via P-NET, together with the error code.
- VIGO receives the data and converts the data into the format used by the PC.
- VIGO hands the converted data over to Delphi, which places it in “X”.

## Set property – Delphi

- In Delphi, the Set routine for an internal object is activated as shown:
- *Object.property:= X*
- The “Set routine” sets the value of the specified property of the specified object equal to “X”.

## Set property - VIGO/P-NET

- After first having used the PhysId to select a remote property in an object located in a slave, VIGO supports the “Get property” in the same way as Delphi:
- *VIGOObject.value:= X;*
- ”Set property”, assigns the value of the selected property with X”.
- When setting PhysId, VIGO converts the object and property identifier into a Swno, using the MIB.

## Set property - VIGO/P-NET

- When VIGO receives a “Set property” with data, VIGO converts the data into the format that suits the slave.
- Then VIGO initiates a “Store” instruction via P-NET, using the SWno to identify the property.
- The slave receives the request, converts the received SWno into a “Set routine” and a pointer to data. The routine is executed and returns an acknowledgement via P-NET, together with the error code.
- VIGO receives the “acknowledge” from the slave and returns to Delphi.

## ”Method” without return value.

- Objects can contain “methods”. Parameters can be passed to a method, with or without a return value.
- In Delphi, “methods” without return values are activated in this way:
  - *Object.method (parameters);*
- When calling the method, the related routines are executed using the passed parameters.

## “Methods” without return value - VIGO/P-NET

- Currently, VIGO does not directly support the notation used for “methods”, but by adding a method to VIGOPro and a new “method Kind” to the MIB, VIGO can be extended to support “methods”.

# P-NET

## “Methods” without return value - VIGO/P-NET

- VIGO will then support the “methods” in the same way as Delphi:  
*VIGOObject.method (parameters);*
- For methods not requiring a return value, VIGO initiates a Store instruction via P-NET, where the parameters are transferred to the method located in the slave as data or a data structure, identified by a SWno.
- The “Method” is a “write only property” compared with the set/get attributes of a "Property". Methods are treated as a “Set routine” by P-NET and by the slave.

## ”Method” with return value.

- Objects can contain a “Method” with a return value. Parameters can be passed to a method, and the method returns a value.
- In Delphi, a “method” with parameters is activated in this way.
- `X: = Object.method (parameters);`
- When calling a “method”, the related routine is executed with the passed parameters and Delphi stores the return value in “X”.

# P-NET

## ”Method” with return value - VIGO/P-NET

- With the earlier described “Method kind”, VIGO can be extended to support “methods” with return value.
- The MIB element must specify two data formats, the format of the parameter structure and the format of the return value.
- After first having used the PhysId to select a remote method in an object located in a slave, VIGO supports the “methods” in the same way as Delphi:

*X: = VIGOObject.Method (parameters)*

# P-NET

## ”Method” with return value - VIGO/P-NET

- The method in VIGO initiates a Store instruction via P-NET, where the parameters are transferred to a method located in the slave, as data or a data structure, identified by a SWno.
- The server provides the acknowledgement “Answer comes later”.
- When the method routine has been executed, the server returns the response as it would for a delayed Load.
- VIGO converts the return value, and Delphi writes it into “X”.
- The parameter length and the return value length are not limited.

## Methods and the P-NET standard

- There is no problem implementing the methods, although the P-NET standard does not describe such a principle.
- There is nothing special for the gateway to do.
- The client and the server only have to know that it is a method, together with the type of the parameter and the type of the return value.
- The implementation of the “method” will therefore not violate the P-NET standard.

## Channel type

- The channel type used in P-NET is another word for I/O object class.
- Within a node, a number of channel instances of the same type/class can be defined.
- Each channel instance has its own independent data.
- Registers in a channel are equal to “properties”.
- Some registers are “methods” without return value.

## Node type

- The Node type used in P-NET is another word for Node object class.
- A Node instance is created when installing a new node, with software of a given Node class/Node type.
- A number of Node instances of the same type/class can be installed.
- Each Node instance has its own independent data.
- Process-Pascal is used to define new Node object classes.

# P-NET

## Synonyms between Object oriented names and the P-NET naming scheme

- Class : Node type, Channel type, Function block
- Object : Project, Node, Channel, Function block
- Property : Register, SWno
- Method : Function
- Get : Load
- Set : Store

## Trends

- The trends within process control and supervision are expected to follow these directions:
  - Distributed interface.
  - Distributed intelligence.
  - Object oriented approach: Channel structure, Function block.
  - Object oriented programming: C++, Delphi etc.
- This means the requirement will increase for " Remote access of objects" via fieldbuses, where the distributed intelligence can be activated through properties and methods.

## Conclusion:

- The easiness of using P-NET is originating from the basic object oriented ideas.
- These qualities are no doubt the explanation for the fast growing interest for P-NET.
- P-NET, a fieldbus, born as an object oriented fieldbus, ready for the future!