The Windows to P-NET Connection in Practice

Chris Jenkins - PROCES-DATA (UK) Ltd.

Introduction

Most industrialists and technologists are now familiar with the use of the PC as a tool in many aspects of the work place and for personal use. One of the reasons for this explosion in the utilisation of PC's has been the intuitive features of software applications, which use graphical operating systems such as Windows.

It was not so very long ago, that an operator may have had to type in keyboard commands such as "Save" or "Print", using a texually orientated screen. Nowadays, most people are familiar with the use of a mouse, to choose an item from a pull down menu, and with the most advanced applications, use the mouse to select an item or press a control button, on a highly graphical screen. There has therefore been a natural progression towards the greater use of such user friendly facilities, to monitor and control activities which exist outside the PC environment, such as within a process or manufacturing plant. In order to achieve these “virtual control room” facilities, a reliable and efficient form of communication must exist between the PC/s and the field devices. During the last 10 years, field communication protocols (fieldbuses) have been developed and extensively used, culminating in the availabilty of recognised and standardised (EN 50170) fieldbus types. P-NET is such a fieldbus, which has been used in many thousands of applications worldwide.

Visual Basic

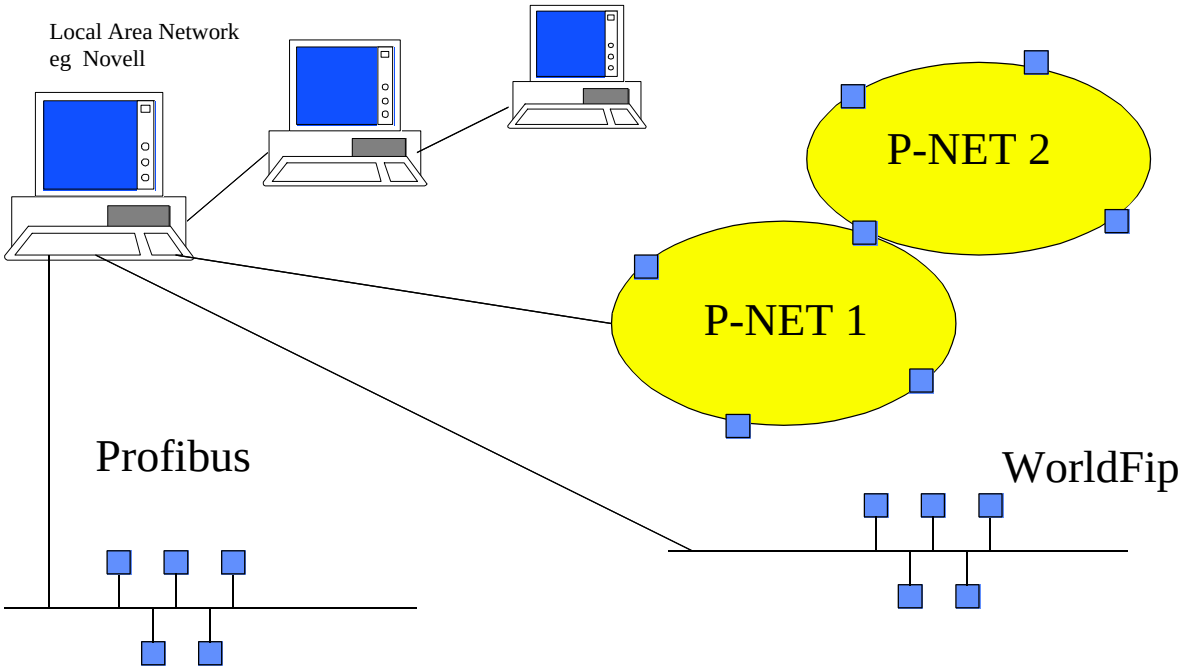
Of course, a lot of these new graphical PC applications use sophisticated progamming languages to produce this user friendly environment. Whilst the most popular language used by professional programmers for writing operating systems, associated drivers, internal libaries and some of the more heavyweight Windows applications, is "C", there are some more user friendly languages such as Visual Basic, which can be used to produce very powerful applications, without having to be a computer guru.

Microsoft have encouraged this concept, by not only producing Visual Basic as an independent application development language for both 16 bit, and 32bit applications for Windows 95, but have also provided it as the “macro language” for their more popular standard packages such as Excel, which is a spreadsheet, Access, which is a database, Word which is a word processor, and Project which is a project planning application. This means that the more lowly programmer or developer, has a path to produce stand alone applications, or configure the powerful standard applications, to suit his own requirements, without having to start from square one.

Furthermore, Visual Basic and VB for applications, which is the name used for the macro versions, all support OLE. This means that messages, data, graphics, sounds and video can be transferred between any applications supporting this technology.

VIGO

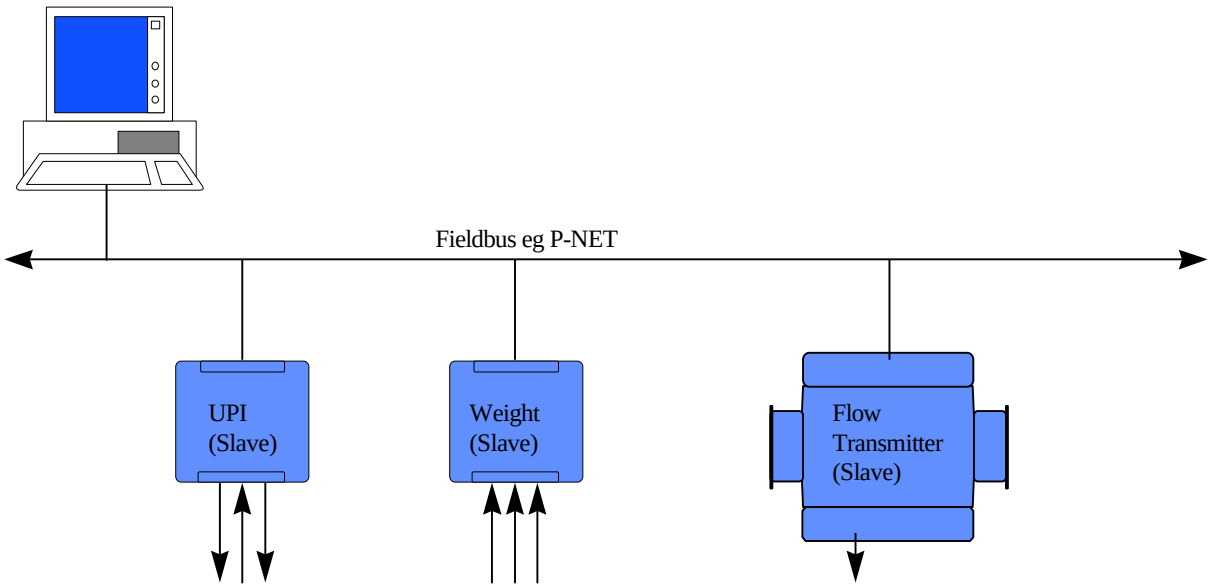
So what has all this to do with the Windows to P-NET connection? Well, VIGO is the key. VIGO is a powerful Windows based application, in the same way that a word processor or spreadsheet is. However, this program does not normally present itself as a highly graphical application, but is a communications program which operates in the background. In some ways, it can be regarded as an operating system just like Windows itself, but rather than dealing with operations of the discs, memory, printer etc., it deals with the transfer of data to and from the PC, with the outside world. In a similar way to Windows controlling data from an application such as Word to a printer, via a printer driver, VIGO does the same sort of thing, except it transfers data from an application to a network such as the P-NET fieldbus or Ethernet, also via a driver. As Windows is able to "talk" to all sorts of printers via specific drivers, so too can VIGO. This means that by adding drivers for other fieldbus and network types, a single application can communicate with all of them at the same time. Here the analogy with a Windows printer ends, since data from one printer of one type is rarely sent to another. With VIGO and an application however, data from one fieldbus type can be sent to another fieldbus type.



Figur 1 - Inter-network Communication

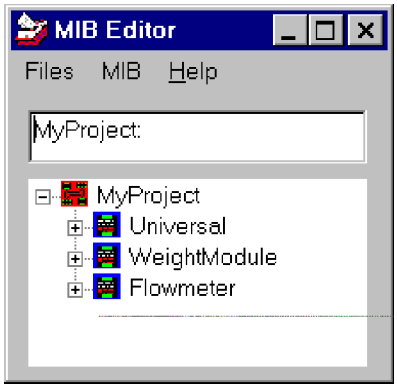
Identifying the Data

So how do we go about designing a Windows application to communicate with P-NET? Well, we need to form a link between a recognisable variable, identifier or object in our program, and a recognisable register name and address in the physical environment. All standardised P-NET channels, such as a Digital I/O channel, an Analogue channel or a Calculator channel consist of a number of addressable registers. A collection of channels are normally manufactured into a stand alone module or P-NET node, such as a 16 channel analogue module or a 32 channel digital module, or a module consisting of a mixture of channels, as can be found in the Universal Process Interface (UPI). Each channel is also identified with a number 0,1,2.... So, to access a physical measurement from say an analogue channel, VIGO needs to know the physical address of the node, the channel and the register in question.



Figur 2 - My Project

Until recently, although not at all difficult, the programmer had to type a list of variable names together with their addresses. But now VIGO provides us with a complete library of standard physical identifiers which are associated with all standardised channels. These can be graphically accessed using VIGO's MIB Editor (Manager Information Base). This has been designed to be very much like a file browser found in the Windows '95 environment, but instead of files, provides an illustative structure of a project, the heirachy of which is project, node, channel, register, sub register... Each name displayed is meaningful in terms of identifying which kind of module, channel and register is being used, but the user can choose to change these to something more meaningful to his particular project if he wishes, by using an alias.

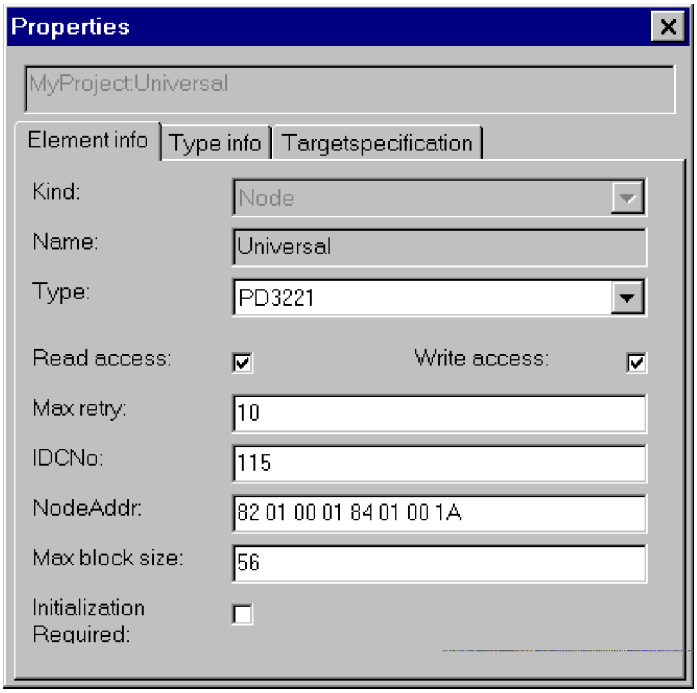


Figur 3

Defining the Project

We must therefore first define a Project, - that is what does our system consist of? We use the MIB Editor at design time, to describe the project in terms of the modules and variables we wish to use or be available. We set up and monitor the properties of these defined objects in terms of their type, address, structure etc.

FIG. 3 shows "MyProject", devised using the MIB Editor, to describe a physical project such as that shown in FIG. 2. The project definition consists of a number of standard P-NET elements, which have been copied from a library of such modules, and which have been given meaningful names. If one of these items is selected with the mouse, and the right hand mouse button is clicked, the properties of this item can be seen. The properties of "Universal" are shown in FIG. 4

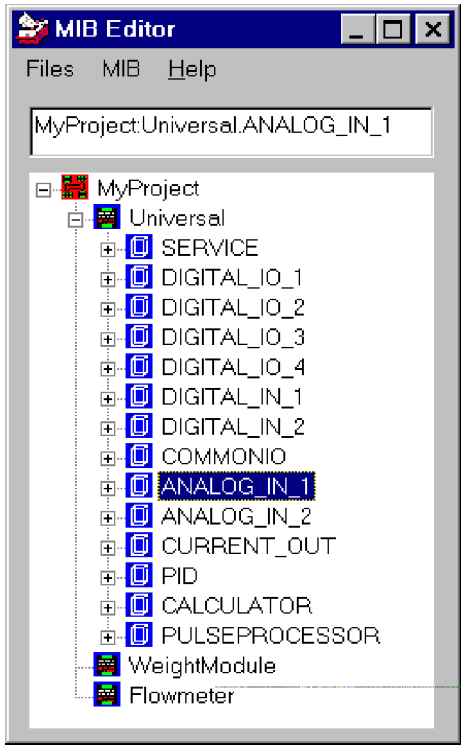


Figur 4

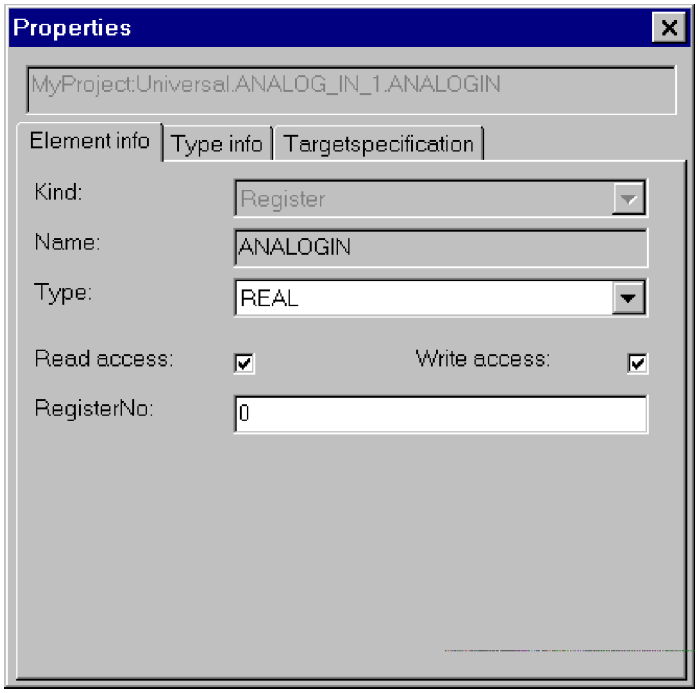
This shows that “Universal” is the name given to a standard module type PD3221 - a Universal Process Interface (UPI). The IDC No defines that this module is to be accessed via P-NET, and the Node Address shows that the UPI has a P-NET address of 1A.

If the browsing facilities of the editor are used, by clicking on the “+”, adjacent to Universal, an illustration is given of all the available channel types within a UPI. FIG. 5

If we focus in on one of these channel types - ANALOG_IN_1, which is one of two anologue channels within a UPI, we can again select the properties window with the right hand mouse button. FIG. 6



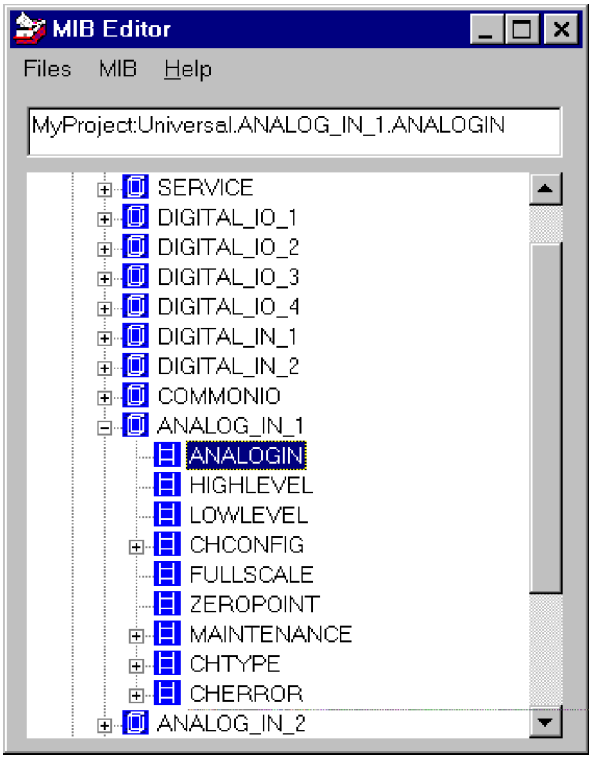
Figur 5



Figur 6

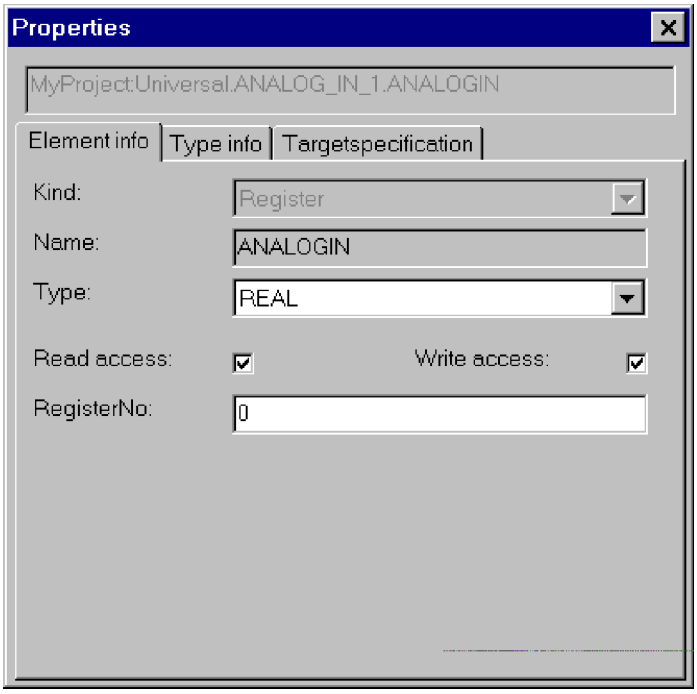
Here we see that this is a standard analogue input channel of type AnalogInCh, which has a fixed SwNo within a UPI of 128 (dec).

We can go even deeper into the structure of the UPI, to see how a particular channel is constructed. For example, this can be done by clicking the "+" alongside ANALOG_IN_1. FIG. 7.



Figur 7

Now we can see the structure of the channel, and again if we focus on the measurement register, the properties of this register can be displayed, by means of using the right mouse button. FIG. 8



Figur 8

Here we see that ANALOGIN is of type real (floating point number), with a register address of 0. Notice also that the full identifier of the register can now be seen at the top of the properties and MIB Editor window. The use of this name in any declaration of a variable within an application, will enable the contents of this register to be used within the program.

Designing the Application Program

Now that we have defined our project, how to we go about designing our Windows application and finilising the connection between the physical project and our program?

Well, to start with, a Windows screen is made up of lots of graphical objects. Probably the most common and familiar graphical objects used are the command button (for control) and the text box (for monitoring). There are many other objects that could make up a window or form, including check boxes, radio buttons, menus, dialog boxes etc. Each of these objects possess properties and methods. For example, a command button has a dimension property, a position property, a colour property, a caption property. Methods associated with a command button define what the button is supposed to do, when say it is clicked or double clicked with the mouse pointer A method is therefore a proceedure or sub routine.

So let us create a Visual Basic Program to show how easy it is to make the connection with a process.

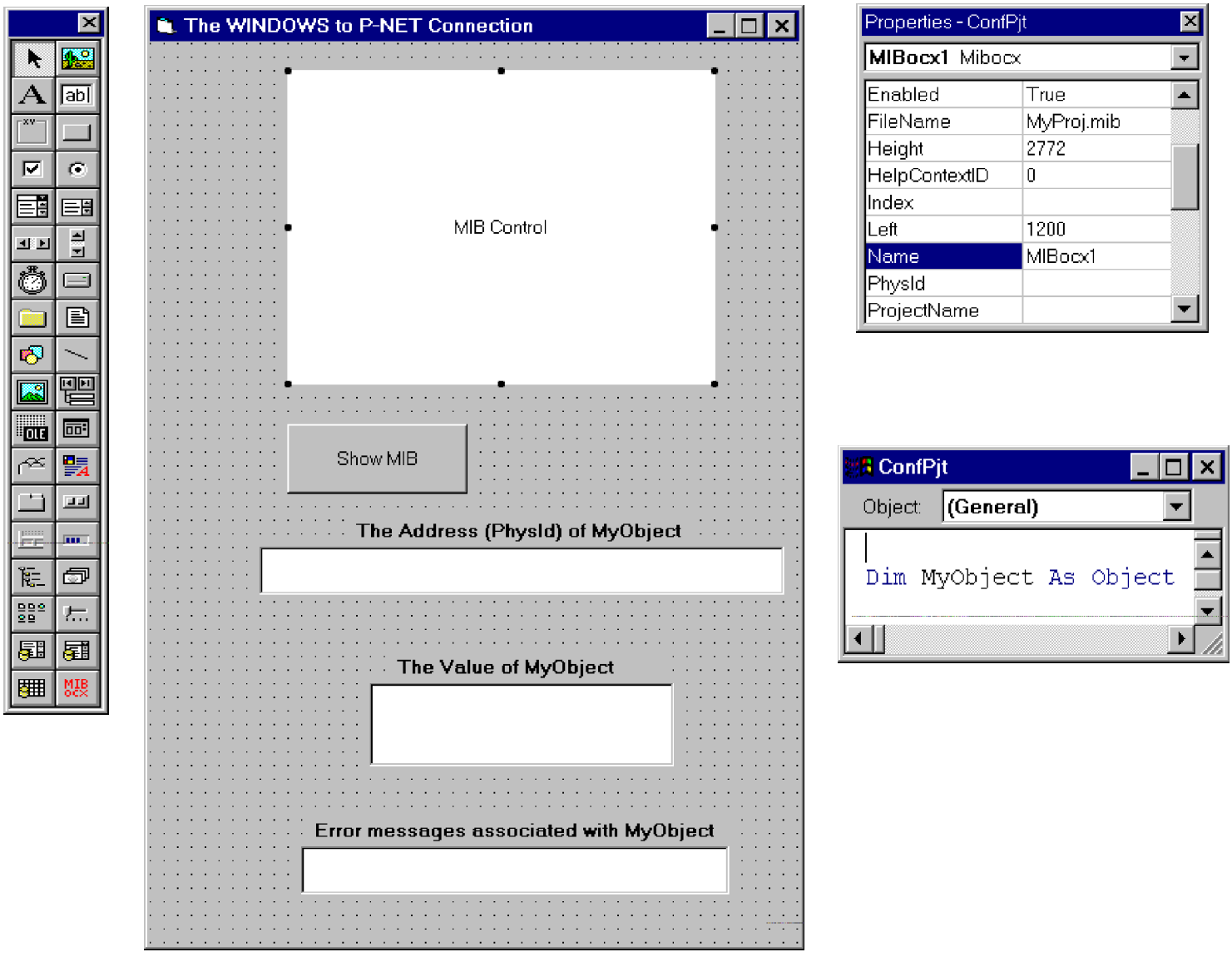


Figure 9

Here I have drawn my program display, using the Visual Basic design screen. This consists simply of a command button, three text boxes and some titling text, to show what is to appear in each of the text boxes. Also included on the display is some illustrative text, which merely describes the small amount of Visual Basic code I have had to write to acheive this versatile application. FIG. 10.

Now, rather than declaring a specific variable from my project, I am using an extremely useful facility provided by Windows, VIGO and the MIB Editor, in order that I can choose the variable I want to display at run time. To do this, I am using a devised OLE (Object Linking & Embedding) control called MIBOCX. This control can be seen in the Visual Basic Toolbox, and allows me to draw an area on my screen, within which I will be able to select and use the MIB Editor, while the program is running. Notice that the design screen provides the facilities for any of the drawn objects to be selected, where the properties and code associated with the selection can be displayed and edited.

Running the Program

Now that the application program has been drawn and coded, it can be run or compiled into an executable file. By clicking the command button “Show MIB”, this starts the link and displays the MIB for MyProject. FIG. 10.

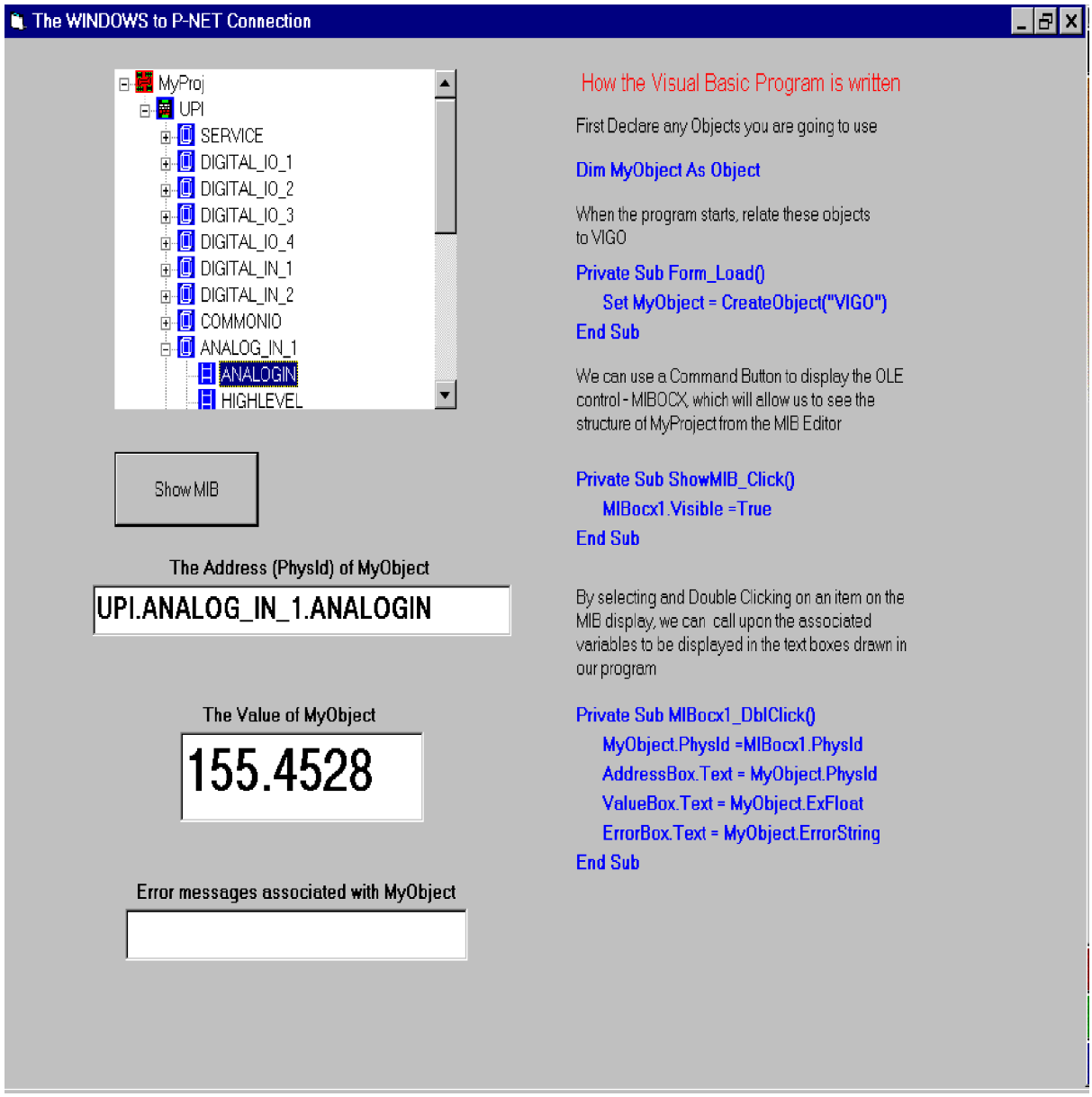


Figure 10

By selecting and double clicking any of the available elements shown by the MIB, the physical identifier of the required measurement register, or other internal register of the same type, will be used by VIGO to request and receive the value of the measurement. Any error occuring during communication is displayed in a text box. The physical identity of the data source is also displayed.

Whilst the code required is extremely compact, this program does provide a versatile, although limited application. However, more objects could be declared, additional text boxes and command buttons added. Perhaps a timed loop could be added so that the mouse button needn't be pressed for each update. According to the code, the value displayed in the Value text box is expected to be a floating point number (ValueBox.Text = MyObject.ExFloat). Other data types could be displayed by replacing ExFloat with ExInteger or ExByte etc. The possibilities are of course endless.

So, there we have it. The Windows to P-NET Connection has been made, with the absolute minimum of effort. Apart from associating additional screen objects with physical objects, it leaves the designer to use his ingenuity to produce other useable applications, using the powerful facilities offered by Windows, VIGO and Visual Basic. Of course, the fact that ready made database orientated application shells such as Excel and Access exist, which use Visual Basic for customisation, means that the same facilities just described can be used with these packages as well, to produce powerful information retreival and data logging applications. But that warrants a separate paper.

Conclusion

So, in order to make the Windows to P-NET connection (or any other fieldbus or network), we need to consider that the link is made up from a few components. Firstly, the physical project, made up of modules, controllers and PC's, each measurement or activation point having a name (identifier) and address. Next, we have VIGO, which amongst other things, provides the communication facility between the outside world and the Windows operating system, and provides the means of describing the project in terms of identifiers and addresses. Finally we have our application program, which by declaring the whole or parts of the project as OLE objects, enables a program variable to be synonomous with a physical variable. This provides a straighforward opportunity for our Windows based PC to monitor and control projects varying from the very simple to the highly complex .