P-NET as a European Fieldbus Standard EN50170 vol. 1

by Christopher G. Jenkins, M.Inst.M.C., F.I.E.I.E.
Proces-Data (UK) Ltd.

Introduction

P-NET was developed in Denmark in 1986 by Proces-Data. The name P-NET is a derivation of “Process Network”, and was designed as a communications link between distributed process control sensors, transmitters and programmable controllers.

Design Criteria

From the outset, the design criteria of the protocol determined that such a network, (now known more familiarly as a fieldbus), should be a highly efficient means of transferring data between "intelligent" or "smart" devices. The principle of distributing processing power throughout the network, rather than centralising the overall control at one point, would significantly reduce the required bandwidth of the bus. Furthermore, by ensuring that any required measurements, set points or configuration arguments transmitted, would already be processed into useable standard data formats, (e.g. a temperature as a 4 byte real in IEEE 754 floating point format), would also optimise network efficiency.

Bit rate was very carefully considered in so far as weighing up the conflicting requirement for data to be transported as fast as possible, but not at such a speed as to negate the use of standard microprocessor UARTS, or restrict the usable distance or cable type.

It was also regarded as important, that all devices should communicate at a single standard baud rate to ensure compatibility between devices, and to reduce any unnecessary additional configuration of hardware or software set up procedures. Working up from standard serial communication rates (1200, 9600, 19,200 ...) it was decided that the optimum bit rate to achieve all the above target specifications, was 76,800 bits/sec.

The initial physical medium chosen, (which continues to be the most popular) was RS485. This also happens to be an international standard, being a multi-drop, balanced line medium, for which standard hardware components are readily available. This allows P-NET to operate over distances of up to 1.2Km, using single screened twisted pair without repeaters. Although P-NET can be used as a single terminated line, it is normally connected as a ring without terminators. This philosophy helps to increase the integrity of the network, in as much as ensuring that it will continue to operate should the cable fail at a single point.

Access to the Bus

The methodology used to control the use of the bus by devices, needed to address at least two questions. The first applied to how many devices were to be allowed to initiate access to the bus. In fieldbus terminology, such devices are called masters. P-NET was designed to be a multi-master fieldbus, and up to 32 requesting masters can reside on the same bus segment. The second question applied to how an individual master would attain communication, without interfering with the transmission of other devices. In order not to affect the efficiency of the bus, an innovative method was devised to ensure that no additional data needed to be transmitted to perform this control. In essence P-NET is a token passing protocol. However, rather than passing an actual data token message between masters, as is the case in some protocol types, sufficient information is present within a normal message for each master to assess whether it has the authority to communicate. This methodology has therefore been called "virtual token passing". Each P-NET message contains information on both the destination and source addresses of the communicating nodes. Each master has a unique node address, and each is configured to be aware of how many masters are expected to reside on the bus. This doesn't necessarily mean that the "Maximum No of Masters" specified will always be present, but provides the means for token location synchronisation. Each master contains two counters. The first, "Access Counter", holds the value of the node address of the currently transmitting master. When a request, followed by an immediate response from a slave, has been completed, determined by the fact that the bus has been idle for 40 bit periods (520us), each of the counters within each of the masters are incremented by one. The master whose counter value now equals it's own unique node address is said to hold the token, and is allowed access to the bus. However, should the incrementation of this counter exceed the value of the "Maximum No of Masters, the counter in each master is preset to 1. This allows the 1st master in the cycling chain to gain access again.

The second counter in each of the masters ("Idle Bus Bit Period Counter") increments for each bit period the bus is inactive. Should any transitions occur, the counter is reset to zero. As explained above, when the bus has been idle for 40

Node Address Field	Control/Status	Info Length	Info Field	Error Det.
2 - 24 bytes	1 byte	1 byte	0 - 63 bytes	1 - 2 bytes

Fig 1 P-NET Frame Format

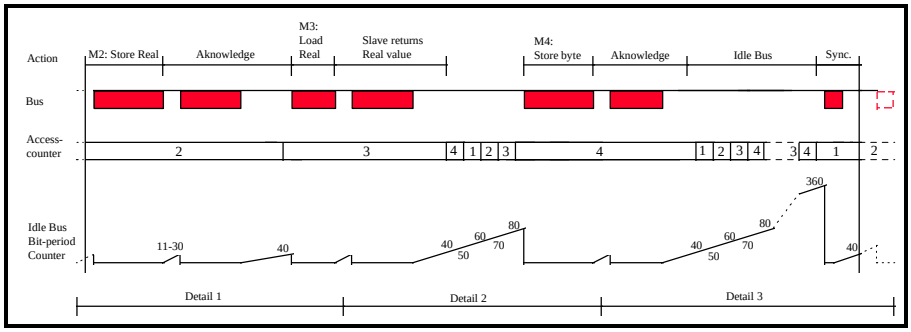

Fig 2 Virtual Token Passing Principle

bit periods following receipt of a slave response, all access counters are incremented by one, and the next master is thus allowed access. Should that master have nothing to transmit (or indeed isn't even present), the bus will continue to be inactive. Following a further period of 130us (10 bit periods), the counter will have reached 50, 60, 70 ...., and all access counters will again be incremented, allowing the next master access. The virtual token passing will continue every 130us, until a master does require access.

Of course a system could consist of just masters (up to 32 per segment), each communicating with each other. However, systems normally consist of both masters and slaves, of which there can be up to 127 devices per bus segment. By definition, a slave is a device which can only respond to a request for information, and cannot independently instigate communication. That is not to say that a slave cannot be "intelligent". The principle of distributed processing power dictates that slaves often possess the ability to perform highly complex and autonomous tasks. The P-NET protocol defines that a slave must produce an "immediate response" to a request to send or receive data. This time frame must not exceed 390us, otherwise a “no response” error will be generated, and the master token will be passed on.

These principles were designed to make P-NET particularly deterministic, and ensures that any particular master does not have any hierarchical priority over any other. Furthermore, it can be deduced that there is no need for any complex arbitration hardware to be considered.

In keeping with the principle of equal priority, a transmission is restricted to up to 56 bytes of actual data, before the token has to be passed on, to allow other masters to gain access. All this means, is that should there be a requirement to transfer large amounts of data between a master and slave (or another master), such as a programme or database, this activity is transparently fragmented into 56 byte data packets. The overall effect is that the rest of network continues to operate normally without being constrained by a single node.

Resultant Speed

The overall combined effect of all the above principles of operation, means that P-NET is capable of completing up to 300 requests and responses for a 4 byte real measurement, per second. These requests do not necessarily have to involve a single master, but can include a combination of every master and slave within a particular system. The cycle time to obtain such a processed variable can be calculated as being less than 3.5ms, which exceeds the speed of many other protocols operating at much higher bit rates. Furthermore, if a four byte message consisted of an array of 32 booleans, 9600 digital states could be monitored per second throughout a system.

Multi-net Operation

Having discussed the principle of operation for a single P-NET bus, it needs to be made clear that P-NET was also conceived as a Multi-net protocol. This is possible because the protocol incorporates variable address length capabilities, which means that the address path defined between a master and a slave can include routing through multi-port masters. The important advantage yielded by such a concept, is that a project (e.g. a process plant or building automation system) can be divided up into a number of logical sections. The consequence of this structure is that although a master can transparently access a slave through one or a number of routers, each individual network operates autonomously. One obvious advantage is that the overall speed of system data transfer is increased, in proportion to the number of nets, in that while one master is communicating with a slave on one network, another master on a

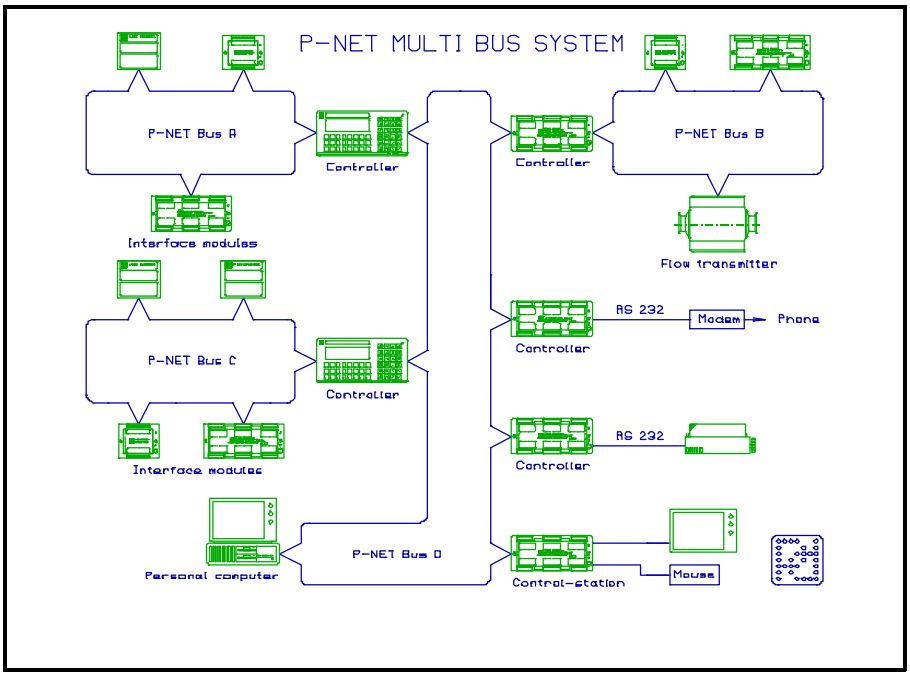

Fig 3 Multi-net Structure

different network can simultaneously communicate with another slave. Other attributes associated with this philosophy are that should one section of the plant be non operational, due perhaps to routine maintenance or power failure, this will not affect the serviceability of the other sections. Furthermore this structure provides a natural network redundancy, in that a number of routes can be physically provided between a master and slave. Thus, if one route becomes unusable, an alternative route can be utilised. This facility becomes particularly important in systems requiring enhanced safety, such as on ships or aircraft.

Distributed Control

The most widely used device within a P-NET system is the slave. Whilst P-NET can be used economically within simple systems (e.g. single master and one or two slaves), the most spectacular examples involve numerous masters (including PC's) and an even greater number of slaves. The reason for this is very much associated with the original concept of distributed control, where a slave is physically located close to the process, or sensing and activation activity. A slave can be a simple sensor or actuator, a specialised transmitter, an autonomous control block, a PLC device or a mixture of all these. The tendency is that masters do not normally possess any I/O, although the protocol does not preclude these facilities. As previously mentioned, a slave can possess a high degree of processing power, and is only inhibited from instigating a transmission. Examples of such slaves include flowmeter transmitters, where all the processing required to provide the system with flowrate, batch and accumulative totals, batch, PI and temperature compensated flow control, takes place within the transmitter itself. The traffic on the bus is therefore reduced, and only needs to be used for reading any of the processed variables required, and occasionally down loading a batch or flow set point, or changing configuration data. Another powerful slave example is the Universal Process Interface (UPI) which includes I/O for 6 digital sensors, 2 analogue sensors, an analogue output, an PID channel, a calculator channel (programmed for PLC type functions) and a pulse processor capable of counting pulses at 200 kHz. Other available slave types include weight, thermocouple, pure digital and pure analogue types.

This collection, together with single and dual port programmable masters, PC interface cards, repeaters, fieldbus management software and compilers has been a significant enabling factor in the implementation over 5000 projects, (May 1996), involving an estimated 50,000 nodes worldwide, since P-NET inception.

Channel Structure

Accessing data from a slave or a master, not only involves the use of the P-NET node address but also the address of the variable required. Rather than using an absolute addressing strategy, all P-NET variables, whether they be of simple data types, such as integer, boolean, real etc, or of complex types such as records or arrays, are mapped into symbolic or logical addresses called Softwire Numbers. This philosophy has been further enhanced to produce the object orientated concept of Channels. A channel can be regarded as an I/O description or profile, and a number of channels can be incorporated into a single P-NET module. If we take a 16 channel digital input/output module for example, the user will know that each digital input or output channel will be identified as $1 to $10. A channel of any type possesses up to 16 addressable registers, also symbolically identified from $0 to $F. In other words, the user is protected from needing to know any absolute addresses the device application program might use, since translation from one to the other is internally mapped via a table controlled by the manufacturer. Any processed measurement variable, whether digital or analogue, tends to always resides in register $0. To access the state of a sensor or actuator connected to one of these channels the user merely needs to address the appropriate channel number and look in register $0. The other available registers within a particular channel are used to hold data associated with the measurement or process, such as set point, alarm limit, count, load current maintenance history, configuration and error data.

One specific rule, which must be adhered to in any P-NET device, is that it must contain a Service Channel. This channel is always identified as Channel 0. The structure of this channel is very specific, in that it contains data appertaining to device identity such as manufacture, serial number, device type, node address. It is also the means for remotely resetting the device and monitoring any device errors. It may be realised that such information, available at a consistently specific address in any device type, provides the basis for "plug and play". For example, “Device type ” will be recognised by a controller or PC and it's complete structure can be mapped within an application. The unique serial number string, embedded within all

Variables within Digital I/O Channel **x** — Channel identifier: **Digital_IO_x**

SWNo	Identifier	Memory type	Readout	Type	SI Unit
x0	FlagReg	RAM Read Write	Binary	Bit8	---
x1	OutTimer	RAM Read Write	Decimal	Real	s
x2	Counter	RAM Auto Save	Decimal	LongInteger	---
x3	OutCurrent	RAM Read Write	Decimal	Real	A
x4	Operatingtime	RAM Auto Save	Decimal	Real	s
x5					
x6	FBTimer	RAM Read Write	Decimal	Real	s
x7	FBPreset	EEPROM RPW	Decimal	Real	s
x8	OutPreset	EEPROM RPW	Decimal	Real	s
x9	ChConfig	EEPROM RPW	----------	Record	---
xA	MinCurrent	EEPROM RPW	Decimal	Real	A
xB	MaxCurrent	EEPROM RPW	Decimal	Real	A
xC					
xD	Maintenance	EEPROM RPW	----------	Record	---
xE	ChType	PROM Read Only	----------	Record	---
xF	ChError	RAM Read Only	Binary	Record	---

Fig 4 Standard Digital Channel Structure

devices, is also the means of recognising and changing the node address on the fly. The fact that symbolic addressing is used to obtain and send data to devices, means that the actual absolute address of a variable is not seen or needs to be considered by the user. However, an important factor to be recognised, is that any manufacture utilising the structure of a standardised channel need not be concerned about where data is held within his application memory, or what type of micro-processor is used. By mapping recognisable Softwire Nos. to absolute addresses via a table, ensures compatibility between devices from different sources.

Process Objects

The concept and use of channels has often been referred to as an additional layer 8 (User layer) within the OSI communications model. This provides the user with an object orientated view of a fieldbus as a whole, and specific measurements or processes in particular. The fact that a catalogue of standardised channels exist, such as general purpose digital and analogue channels, PID, weight, calculator, program, communication, printer, power, etc, means that a user can conceptualise particular measurements or processes as a single entity, and that manufacturers of new devices have a standardised structure on which to base their equipment.

The use of PC's and networked workstations within systems is now so common, as to be a standard element within all but the most simple P&ID. PC's have always been the basis for P-NET master and slave program development and device configuration. However, the object orientated nature of P-NET provides a natural and powerful alliance with operating systems such as Windows 95 and NT. Standard Windows applications such as Excel and Access provide an easy basis for incorporating fieldbus variables within these structures. Furthermore, object based development languages such as Visual Basic, Delphi and Visual C++, provide an application designer with all the tools necessary for producing a wide variety of MMI, Management Reporting and SCADA packages. All this is possible because of OLE2 Automation technology. However, this technology is not real time based, and a fast, object orientated link is required between the real time aspects of the fieldbus and the OLE message passing facilities offered by Windows. A real time, Windows based Fieldbus Management System has been developed to provide this link, not only with P-NET, but to also provide simultaneous communications with other protocols. This real time 32 bit "operating system", called "VIGO" (Virtual Global Object), enhances Windows, to enable fieldbus variables to be included within applications, as if they where internal variables. The facilities offered by VIGO enable a user to visually structure and manipulate a project using devices, much in the same way that Windows offers a Device Manager for internal hardware. Furthermore, individual channels can be regarded as objects containing associated properties, to which methods can be applied.

Conclusions

The concept of P-NET has always been to provide a fast and efficient, non hierarchical, single speed local area network, for field intercommunication. Due to this philosophy, consideration of "high speed backbones" or hierarchical changes to the basic protocol, have been unnecessary. Furthermore, the design of new slave equipment doesn't require the use of special chips, which have to be hardware and software linked to the application microprocessor, since the protocol can be embedded within the application code. However, there is a recognition that workstations also need to be networked together, and there is often a requirement for real time derived data to be transferred between them. VIGO also provides these facilities through Windows. Thus variables originating from within a plant can be distributed between workstations and servers via any common LAN or WAN.

There is also a recognition that many other standard and proprietary fieldbuses, device buses and sensor buses exist. There are therefore bound to be situations where intercommunication between incompatible protocols will be necessary or even desirable. VIGO provides the ability for additional protocols to be

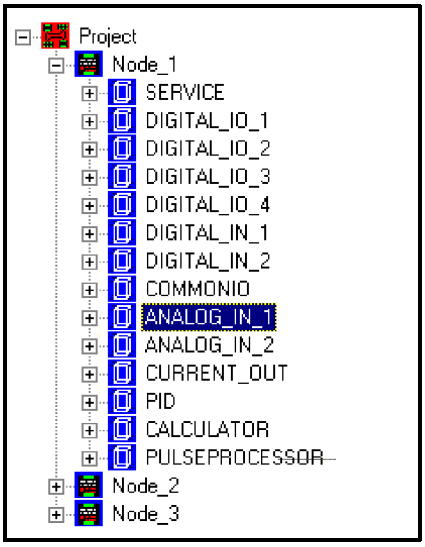

Fig 5 VIGO Browser Window showing module/channel structure of example project

dynamically loaded. Thus, the PC can be used as a gateway between these various types, where a common application can utilise, control and interchange measurement data between these types.

The design philosophy behind P-NET was laid down some time ago. It has not evolved as a consensus based protocol, nor has it been modified to incorporate a mixture of protocols. National standard status was achieved as recognition of it's wide and proven use. The subsequent adoption of P-NET within the new European Fieldbus Standard as EN 50170 Vol. 1, stands as an indicator that P-NET not only meets all required design criteria for inclusion, but also provides users with a choice of standard fieldbus types, to suit their own particular operational requirements.