Produced by: The International P-NET User Organization 590 004 02

The P-NET Fieldbus

for Process Automation

P-NET
- the European Fieldbus Standard
EN 50170, Volume 1

Applied P-NET Applications

Textile Production Plant

Test Line at Bosch-Siemens Dish Washing Machine Factory

Applied Building Automation

Animal Feeding & Climatic Control

P-NET in General

The P-NET Fieldbus is designed to connect distributed process components like process computers, intelligent sensors, actuators, I/O modules, field and central controllers, PLC's etc., via a common two wire cable, as shown in fig. 1.

This replaces traditional wiring, where a great many cables are involved.

Process data (e.g. measurement values, valve signals) are transmitted digitally. P-NET is also used for data collection, for configuration of nodes/sensors, and for down-loading of programs.

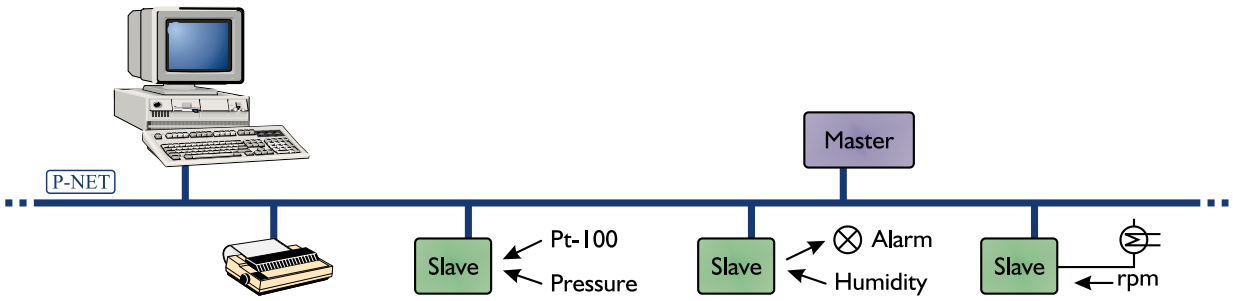


Figure 1: P-NET Fieldbus with distributed process components.

Apart from the usual measurement values and status data, the bus provides a bidirectional exchange of additional information concerning limit values, actuator positioning and feedback signals, fault signals and internal system data.

P-NET can be used to download parameters and programs to modules, which then control the process. The use of intelligent P-NET sensors and actuators also offers much better diagnostic features than with traditional wiring.

Further comparisons with conventional wiring, show that the incorporation of P-NET offers proved advantages when applied to industrial processes.

The result is a simplification of planning and installation, a reduction in the amount and cost of cabling, a reduction in installation and maintenance costs, a reduction in installation errors, leading to a more straightforward future expansion of applications. Instant information about field device faults, and faults in the cable, can be detected automatically by the network protocol.

P-NET applications are characterised by their low cost for a small system. The cost rises linearly with the size of a system.

P-NET is as well suited for small plants, as for large plants having many controllers, sensors, and interface modules. In addition, any such system is always ready for any necessary expansion.

The History of P-NET

P-NET was conceived in 1983. The first product using this multi-master Fieldbus was launched in 1984.

The multi-network and multi-port functions were added to the protocol specification in 1986. The first operational P-NET multi-port product was produced in 1987.

The P-NET standard became an open and complete standard in 1989, for adoption worldwide.

Due to an increasing interest in P-NET, the International P-NET User Organization was formed a year later.

Application Areas

The P-NET Fieldbus has been used for many years, and more than 5000 applications are now in operation worldwide. Applications range from simple installations with a few I/O points, to very large and complex installations using many thousand I/O points.

P-NET applications are found in the process industry environment and in discrete parts manufacturing plants.

The following typical examples show where P-NET is currently installed and running: Dairies, breweries, environmental control in agriculture, animal feeding systems, asphalt and concrete production, textile industry, milk/oil/fertilizer distribution trucks, quality control systems, power plants, solar power plants, plastic moulding, ship engine control, tank management/alarm systems (approved by German Lloyd, Bureau Veritas, Norske Veritas, Lloyds Register of Shipping), data acquisition, water supply, building automation, fuel management systems, (approved as legal for trade by PTB, NMI, NWML, ...).

The typical P-NET application requires response times measured in ms, and a bus length up to one km or more. There are other types of applications which demand a response time measured in ìs. For these applications P-NET is not appropriate.

Principles of P-NET

The electrical specification of P-NET is based on the RS485 standard using a shielded twisted pair cable. This allows a cable length of up to 1200 m without repeaters. Data is sent as an asynchronous transmission in NRZ code.

P-NET interfaces are galvanically isolated, and up to 125 devices per bus segment can be connected, due to a special clamp circuit, and again without the use of repeaters.

P-NET is a very efficient Fieldbus protocol, in that it can handle up to 300 confirmed data transactions per second, from 300 independent addresses.

Data can be transferred in the form of fully processed values (floating point), such as temperature, pressure, current, voltage etc., or as blocks of 32 independent binary signals, indicating valve states, switch positions etc.

This results in a performance of up to 9,600 binary signals per second being accessed from anywhere within the complete system.

This high rate of fully acknowledged data transmissions can be achieved, because P-NET slaves handle the processing of data and the reception or transmission of frames, in parallel. The processing of a request by the slave is initiated as soon as the first data bytes arrive. This is in contrast to dedicated chip solutions, where the entire frame arrives before processing begins. In this way, the standard P-NET data rate of 76,800 bit/s, is not a limiting factor in performance.

The performance can be compared with systems using data rates up to 500,000 bit/s. See a detailed description on page 17 “P-NET Compared with Dedicated Fieldbus Chip Solutions”.

P-NET is a multi-master bus, which can accept up to 32 masters per bus segment. All communication is based on the principle, where a **Master** sends a request, and the addressed **Slave** returns an immediate response. Requests can be of a read or write type. Masters and slaves are shown in fig 2.

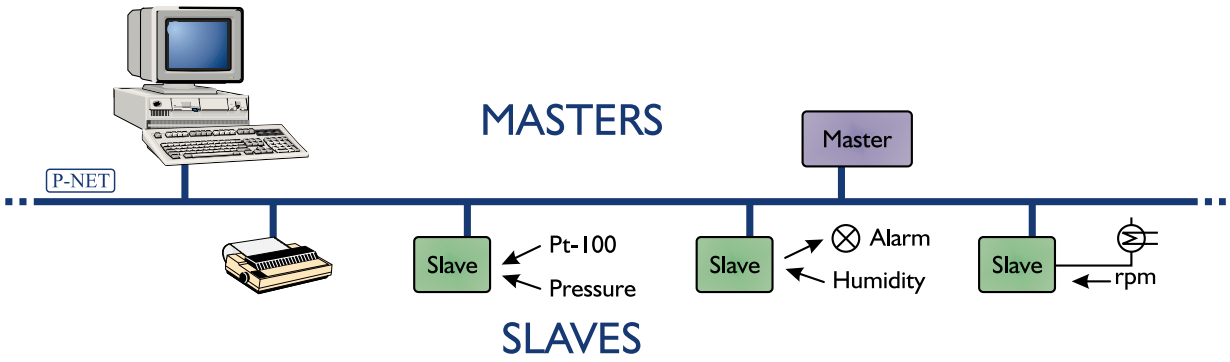


Figure 2: Masters and slaves on a P-NET Fieldbus.

Data transferred on the bus can be of a simple or complex type, to satisfy the requirements of measurement and control. Simple types include boolean, byte, char, word, integer, long integer, real, long real and timer. Complex types include array, string, record and buffer.

The data format is a part of the P-NET standard.

The right to access the bus, is transferred from one P-NET master to another, by means of a token. P-NET uses a method called “virtual token passing”, which does not require messages to be sent over the bus.

When a master has finished bus access, the token is automatically passed on to the next master, by a cyclic mechanism based on time. The method used in P-NET differs from that used in other multi-master systems.

Other busses such as Profibus for example, use real message telegrams for transferring the token. This results in an increase in master processing time, and reduces the capacity of the bus.

The virtual token passing principle also accepts that a master might not even be present. In this situation, all devices, including other masters, will continue performing normally. See page 16 “Virtual Token Passing” for a detailed description.

Multi-net Structures

The previously accepted way of designing a network architecture for a factory, was to have the Fieldbus directly connected to the sensors and actuators. The Fieldbus would then be connected to a cell-controller, and a number of cell-controllers would be then connected to a cell-network, and so on, up through the hierarchy, ending with a high speed backbone network. The data rate for the network on the next level up was assumed to be a magnitude higher than on the lower networks.

This was perhaps a reasonable philosophy in the past, where all data had to eventually end up in a powerful computer at the top level. The technique for today and the future,is to distribute intelligence between the cell-controllers, interfaces and sensors. At each level, the data becomes concentrated and regulating loops are typically closed within the same bus.

The need for a fast data rate at the higher levels is now decreasing, as more intelligence is distributed. This is the reason why P-NET may be used on several levels in a complete factory automation system.

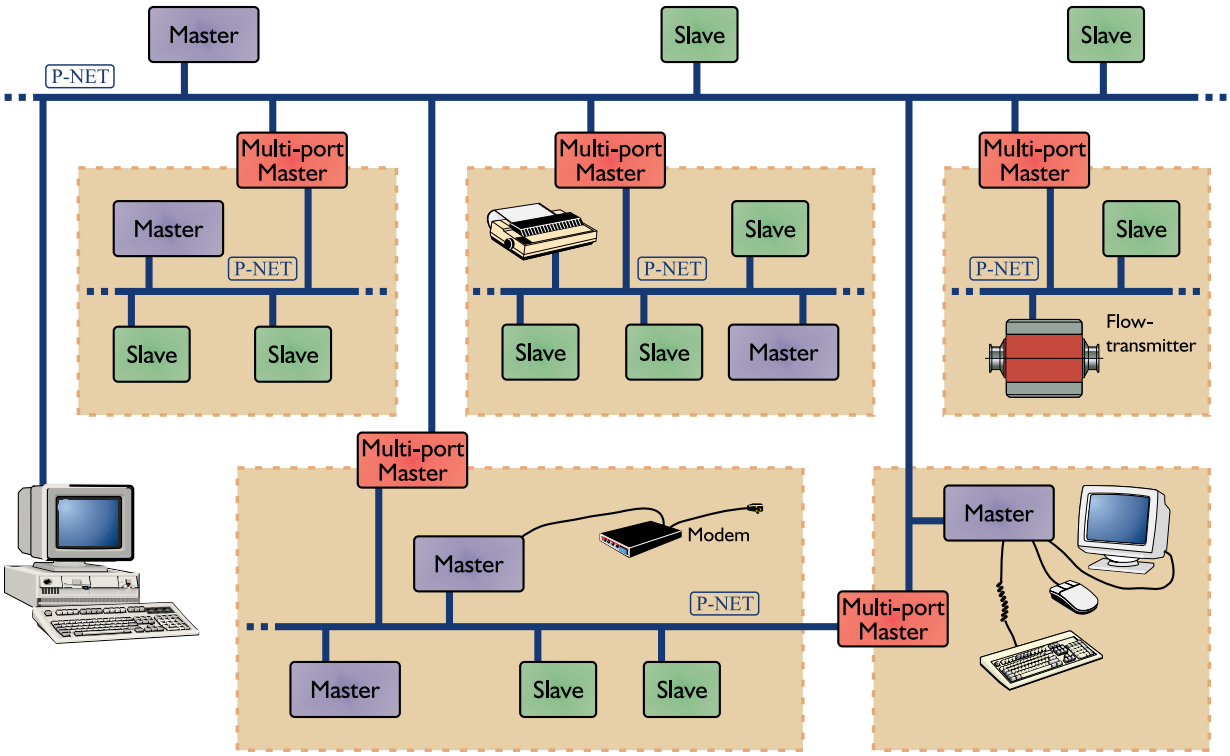


Figure 3: A Multi-net structure with the P-NET Fieldbus

Dividing a system into cells, corresponding with each section of a plant, makes it possible to shut down a single section without affecting others. Program execution may be distributed in one or more independent processors per cell.

A software or hardware error in one cell, would not affect the others. An individual cell now only has a limited need to exchange data with other cells, e.g. to start and stop processes, to load recipes, to transfer production data etc.

In systems with real distributed intelligence, additional processing power can always be added in the form of additional master controllers. It is therefore possible for a system like this to be expanded.

Among the available Fieldbus systems, only P-NET allows direct addressing between several bus segments, also known as a multi-net structure. This feature is a specified part of the P-NET protocol, and it can be built into the standard operating system of multi-port masters. A multi-net structure is illustrated in fig. 3.

Communication is directed through the different bus segments via nodes with two or more P-NET interfaces. This means that any master on one bus segment can transparently access any node within any other bus segment, without the need for special programs in the multi-port masters. See fig. 4.

The segmentation also makes it possible to have independent local traffic on each bus segment, which increases the update rate and the data throughput throughout the total system.

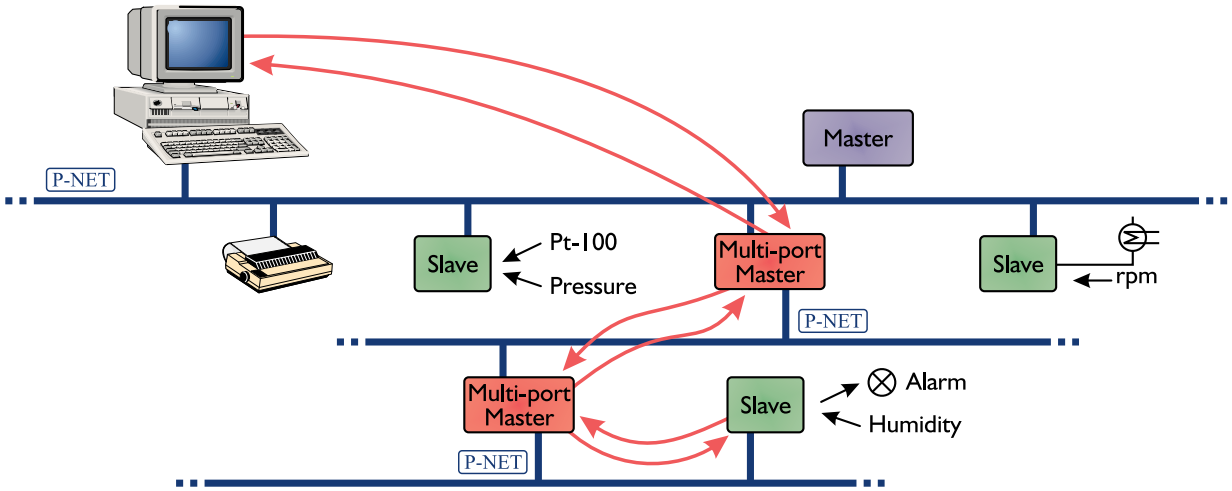


Figure 4: Transparent access through multi-port masters to other bus segments.

The benefits gained by dividing a system into smaller sections are highly significant, because it limits the consequence of an error, to a single segment, which gives higher system security. Furthermore, these multi-net features provide a natural redundancy, which makes the total plant installation very robust with respect to errors. See also fig. 3. An important advantage of the P-NET multi-net topology, is that there is no need for a hierarchical structuring of the bus segments. This is of great benefit when expanding existing P-NET installations, and when coupling to other networks.

An attempt to connect two segments within one node, using a bus system **without** this multi-net facility, requires a special program in that node. Such a program needs to collect all the data from all devices in one segment to make it available to the other segment, which is known as creating “process images”.

With the large amount of data that are available in today’s intelligent nodes, it is almostimpossible to update and maintain a true “process image” for a complete bus segment. Such a procedure occupies a significant percentage of the bus capacity and requires a large amount of memory. Furthermore, it is expensive to create and test a dedicated program for each segment connection.

P-NET does not require such complex “process images” to be built.

Advantages of the P-NET Protocol

All nodes that conform to the P-NET standard can be directly connected to the bus and will immediately communicate together, because P-NET uses only one data rate, and only one choice is given for each of the communication layers.

This differs from other standards, which allow many variations on each layer, resulting in many variants that are not able to communicate together.

Any P-NET module, including a master, can be powered down or connected to or disconnected from the bus, without interfering with the rest of the bus system.

Consequently, modules can be exchanged during system operation, and a system can be expanded while the remaining production system continues to run.

The need for configuration of communication parameters in P-NET is much reduced compared with other systems. In slave modules, the P-NET system integrator only has to set the node addresses, and in master modules, he only needs to define the node address and the number of masters.

Therefore, training is reduced and allows any qualified technician to understand and install a P-NET system.

The distributed processing power of a system can be increased, by simply connecting additional masters.

Special procedures have been included in the P-NET standard, making it possible to change the address of a single node on the network, by means of its unique serial number. This allows individual P-NET node addresses to be changed while the system is still running.

Dip switches and other mechanical mechanisms can be avoided, and it is therefore possible to build hermetically sealed P-NET nodes (e.g. IP-67).

When designing a new device for use with P-NET, benefits will be seen from the fact that P-NET can be used to access any logical or physical address within the device, decided upon by the manufacturer. When a device is implemented with P-NET, both the test procedures performed during the development phase for the application program within the device, and the calibration and maintenance procedures used in the future, can be simplified. P-NET can therefore be used to look inside the device in order to monitor program variables.

The result of a measurement made by a slave, is presented to a master in a pre-processed form, in SI (metric) engineering units. The benefit is significant, since no repetitive scaling or conversion needs to be done by the master(s), leading to considerable savings in processing power. For example, a temperature measurement will be converted to a floating point value by the slave (IEEE 754 standard), and will be presented to all masters requesting the data in degrees centigrade.

Identifiers used for accessing the physical variable on the network, are mapped via a ‘SOFTWIRE’ list. This list is generated while the application program is being compiled. Therefore, no real time translation is required, leading to very fast data access.

To ensure real time data collection, each frame transmitted on the network is restricted to 56 data bytes. If the requested data length is higher than 56 bytes, it is automaticallydivided into several successive transmissions.

Intelligent P-NET Modules

Typical P-NET slave modules give the system integrator more than just Input / Output functions. They very often contain additional process oriented functions varying from simple limit switch monitoring, to PID regulator or program channels, allowing the system integrator to configure local control loops or specify process steps.

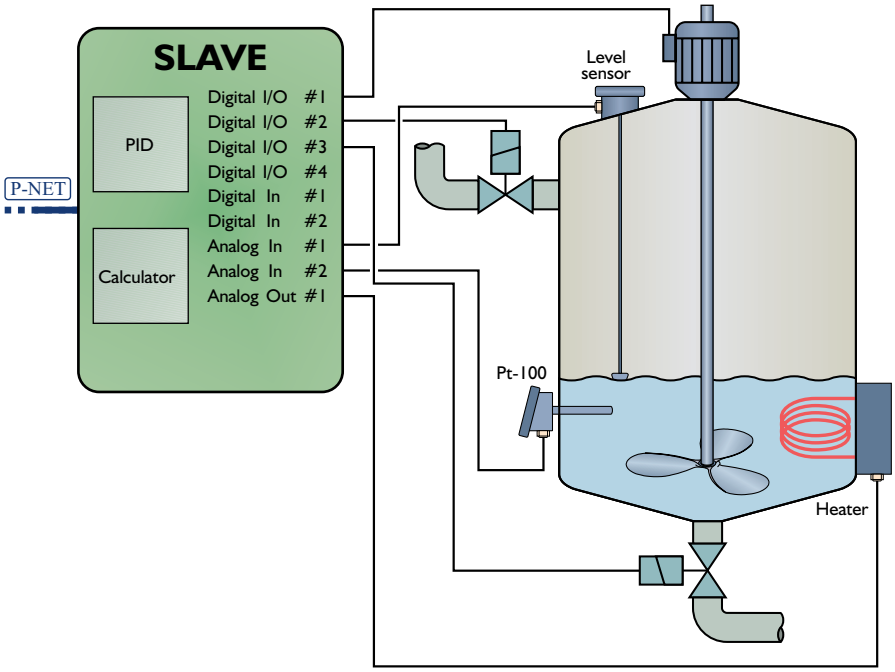


Figure 5: An intelligent P-NET module in a chemical plant.

The diagram in fig. 5 shows how a standard P-NET I/O-module can handle the temperature control, and the loading and unloading of products for a heating vessel within a chemical plant.

In this example, the internal process functions of the module take care of temperature and level regulation, and the control of filling. Only the set points for temperature and level are required from a P-NET master.

Another example of a slave module could be a weight transmitter, where the analogue signal from a load cell is continuously converted, scaled and stored, within the memory of the slave. When a request is received from a master, the slave immediately responds with the latest stored result. Error checking is also continuously performed within the slave, and the master is notified if any error has occurred, by a code in the response message, when the slave is requested.

“Layer 8”: P-NET Channel Structure

Typically, a P-NET Fieldbus device is a sensor, an actuator or an interface module. It can relate to one or more process signals, i.e. a digital output or an analog input. Each process signal is associated with additional information, apart from just the state or the value of the signal. These variables, which are related to the process signal, deal with specific functions for configuration, conversion, scaling, filtering, error messages etc.

In P-NET, this collection of related variables and functions for a single process signal is regarded as a **Process Object**, and is called a **Channel**.

A Channel contains all the necessary data to support the required control functions for the process object. It also includes support for maintenance and technical management of the plant equipment.

A Channel is structured as 16 registers, each having their own relative logical addresses, called SOFTWIRE numbers (SWNo).

These 16 variables or constants within a Channel, can be of any type, including complex, and can be located in different memory technologies.

In order to give a specific example of a standard interface Channel, a Digital I/O channel is illustrated in fig. 6.

Such a Channel can be configured for various functionality, including automatic functions. These functions are input, output, one shot output, timer output etc.

The function is selected by setting a code in the ChConfig register. When the output is configured for timer functions, the preset registers SWNo x7 & x8 are used.

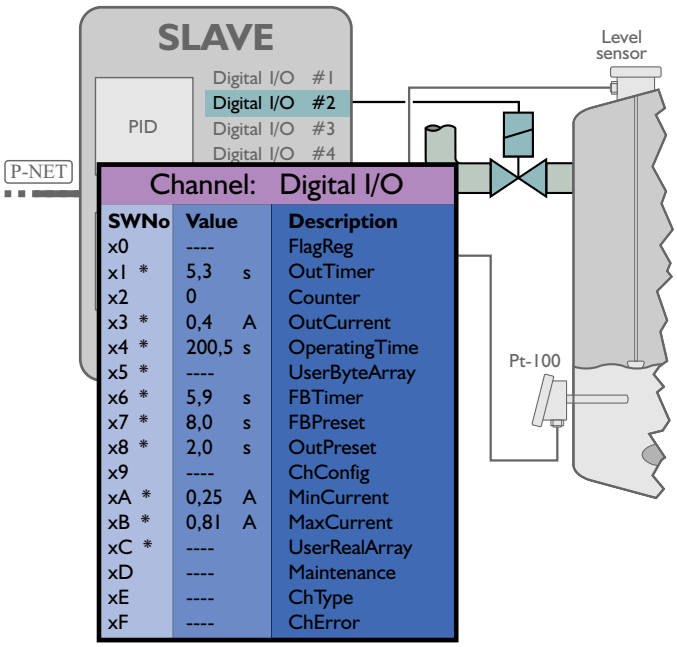


Figure 6: The channel structure of a Digital I/O.

While the input/output pin is active, the OperatingTime register measures the time, and a transition on the pin will increment the Counter, to record input or output activations.

The current in the output load is measured, and can be read in SWNo x3.

MinCurrent and MaxCurrent can be used as a kind of feedback signal to see if the load is connected, and to protect the output and the load.

The Maintenance register can hold information about when and how the last maintenance was performed for the connected valve. The register called ChType must be present in all channels. It is a Record, consisting of a unique number, which defines the channel type, plus an array of boolean, indicating which registers are implemented.

The registers marked “*“ are not mandatory, and may be declared as unused.

One of the important features of P-NET, is error message handling. Therefore, each Channel has an Error Code register called ChError. This register contains error information related to the Channel and the values in its registers.

Examples of errors include Overload, Signal disconnected etc.

A channel need not only deal with process signals, but can also apply to other kinds of data, such as internal function-blocks with corresponding parameters.

An example of such a Channel is a PID regulator, where the output values are the result of a calculation. Other standardised channel type examples include a printer channel, a Communication channel, a program channel etc.

This means that nodes can have different I/O structures. For example, one node might have 16 Digital I/O + 2 analog I/O channels, and another node, 8 Digital I/O + 4 Analog I/O channels, but each single I/O of equal type will be seen as the same by the master, no matter what kind of node it is part of.

The **Service channel** is an important standardised channel type, which must be included in all nodes, whether this is a complex collection of different channels, or just a simple sensor.

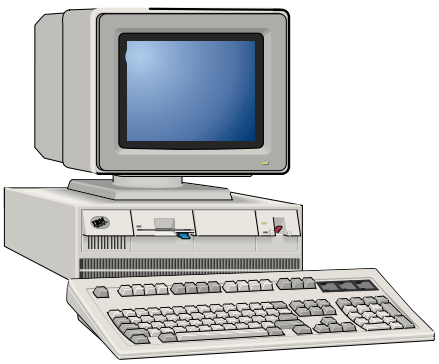
This channel holds information about the node address, serial number, manufacturers identity, overall node error data, and any other data associated with the node. This channel always has a SWNo of 0, and access to the service channel is therefore the same for all nodes. This channel is also used when identifying an unknown node.

The resultant advance of this channel standardisation philosophy, is that from a P-NET master point of view, each channel can be seen and treated in the same way, no matter who manufactured the device, or in which node it is located.

The standardisation also makes it possible to write general programs, which can beused to configure such channel types or to read or write into the registers, independent of the channel's location.

Access to P-NET from PC's

PC's are often used in P-NET installations as one of the masters. PC's are normally connected to P-NET by means of plug in cards.



A product called VIGO has been developed for P-NET.

VIGO is a PC based Fieldbus Management System. It enables a physical plant to be described in terms of data, related data structures and where data is located.

VIGO is also a communication system that manages data security and integrity for data enquiries made within the plant.

VIGO keeps track of the relationship between the physical objects within the plant, and the associated Fieldbus nodes. It also includes the set of files describing the related control programs, configuration and calibration parameters, as well as tools for configuration, backup, download etc.

The routing and handling of several simultaneous information packages for the same, or different networks, are also managed by VIGO, via a real-time communication kernel. When several applications try to access the same bus system, problems will occur in a Windows multi-task environment. This is solved by VIGO, which ensures that communication packages and messages do not get mixed.

A facility available to enable the fast real-time exchange of data between MS-Windows applications is called “OLE2 Automation” (Object Linking and Embedding).

VIGO is an OLE2 Automation Server, which creates a consistent and transparent interface from the user program (application), to the physical elements (objects) within the plant.

In this way, VIGO provides a simple interface to standard program packages such as Visual Basic and Visual C++, spreadsheets, databases, Man-Machine interfaces and other visualisation programs such as SCADA.

Below is shown a Visual Basic program example (EXCEL macro language), using three easy steps:

Step 1: ***set AA = Createobject(“VIGO”)***

VIGO creates a virtual object: AA, which then becomes part of the Visual Basic programming environment

Step 2: ***AA.PhysId = “Setpoint”***

The virtual object AA is made to point to the physical object, by assigning the physicalidentifier to a property called *PhysId.*

Step 3:

X = AA.ExFloat (Get Setpoint)

or ***AA.ExFloat = 37,0*** (Set Setpoint)

All manipulation of the physical object is performed via the virtual object. See fig. 7.

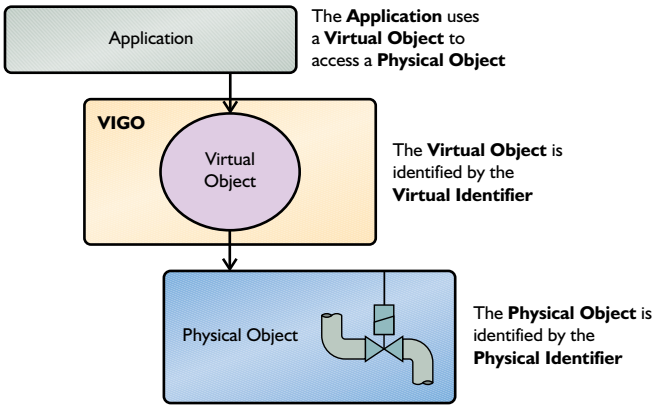


Figure 7.

To operate on the object, a type property must be appended to indicate the type of variable in the node. Exfloat indicates that the object variable is of a real type (floating point), Exbool would indicate a boolean type, etc.

The object can be used in normal assignments, such as set or get functions. Many objects can be created for several independent applications.

VIGO is a collection of several program elements. It is an open system as regards allowing the addition of elements for networks from other vendors. All these elements are handled by and integrated into VIGO, leading to a very simple and well defined interface to any Fieldbus data. The elements of VIGO are shown in fig. 8.

Application: An independent user application program, which communicates via VIGO.

VIGOSERV: An OLE2 Automation server that provides the interface between VIGO and the applications.

IDC: An Instruction/Data Converter, which converts a VIGO service into a specific network instruction, and sets up data using the correct syntax suitable for the destination node, and vice versa.

HUGO2: A real-time communication kernel that provides the possibility of executing several Fieldbus communication applications in parallel.

The Manager Information Base: The MIB contains a set of data structures, which describe the physical system. The MIB translates a variable name into a specific node address, variable address, type specification, etc.

The Drivers: These take care of sending and receiving information via a specific network.

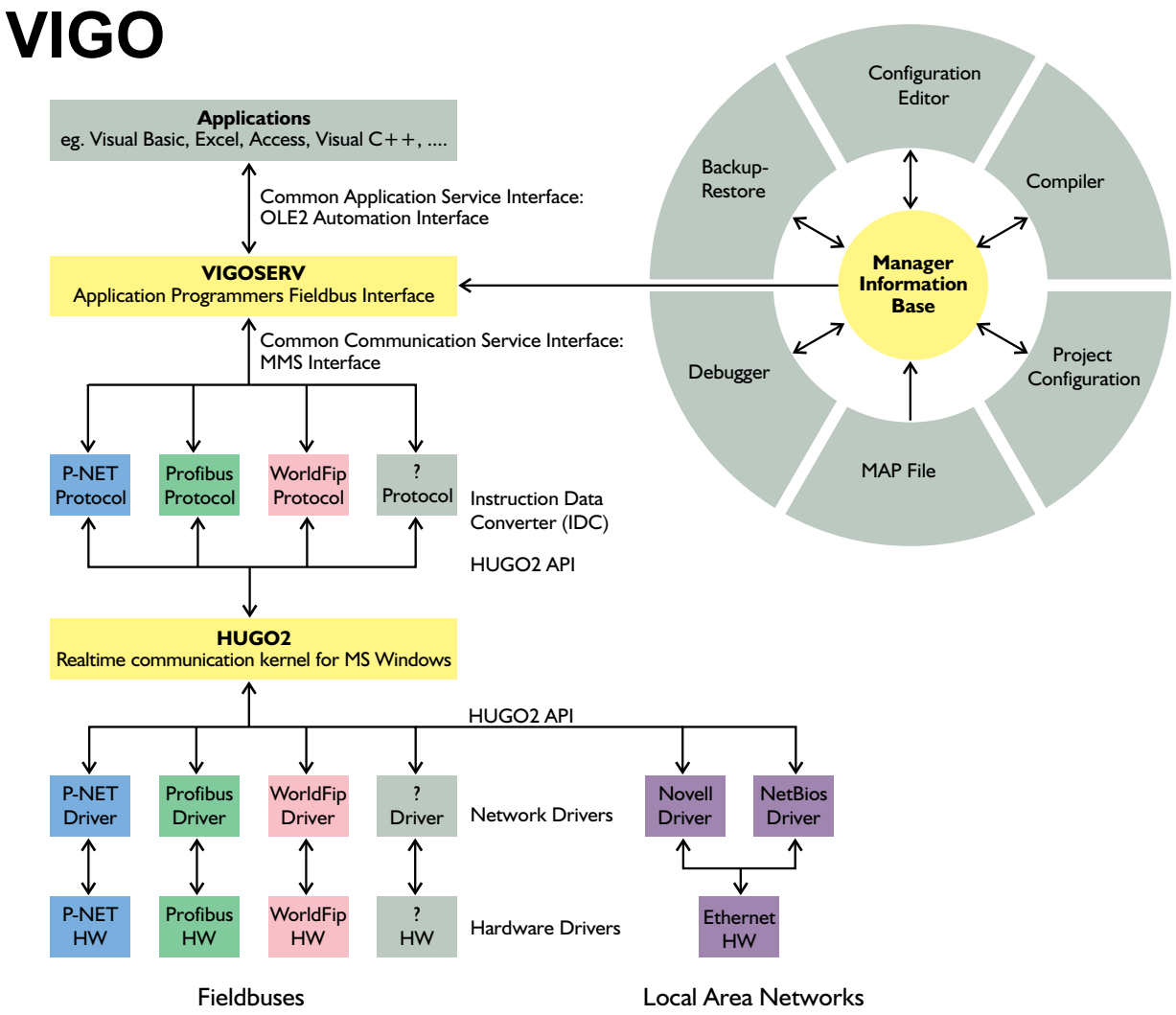


Figure 8: The elements of VIGO.

Software

Besides the standard OLE2 automation method for exchanging data under MS Windows, P-NET drivers for DDE (Dynamic Data Exchange) also exist.

Software tools for monitoring and debugging, Graphic Control Systems, tools for down loading programs and for configuration, editors etc. are all available for P-NET.

Process-Pascal is available as a programming tool for P-NET controllers, which isStandard ISO-Pascal with additional facilities for declaring variables on the network,

and for task management in a multi-tasking environment. Programs written in Process-Pascal use global P-NET variables as if they were local variables. The only difference is to be found in the variable declaration technique. Multi-tasking facilities are also included in Process-Pascal, providing up to 64 tasks in each master.

Ease of P-NET Implementation

One of the reasons for the high number of P-NET installations now operating, can be related to the low cost of node implementation.

The principle of P-NET, is to use the same microprocessor to control the main task of the node (the application), as well as the communication task. Data is only stored in one location. By incorporating P-NET as an integrated part of the device, P-NET can be used to perform configuration and to read the status of the device.

Typically this means that dip-switches for selecting a baud rate and setting the node address can be avoided. See fig. 9.

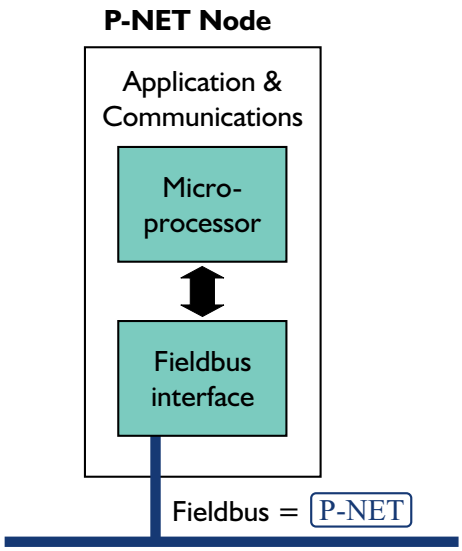


Figure 9: P-NET implementation.

Other Fieldbus types use an add-on circuit in each node, in the form of a separate chip / microprocessor for communication. Data is exchanged through a dualport RAM. This principle always results in a significantly higher cost for the final product. See fig. 10.

There is **no need** for a specific chip-set when implementing the P-NET protocol, because the P-NET communication program for a slave requires only a few kbytes of code. This provides the opportunity to use a common standard single chip microprocessor, which includes a UART. e.g. H8-300, 68HC11, 6805, 80851, 8051 etc.

It can be concluded therefore, that a P-NET Fieldbus node need be no more expensive than traditional microprocessor equipment, having no Fieldbus connection.

Many years of experience have been gained in the implementation of P-NET nodes, and assistance is available for manufacturers, through the International P-NET User Organization.

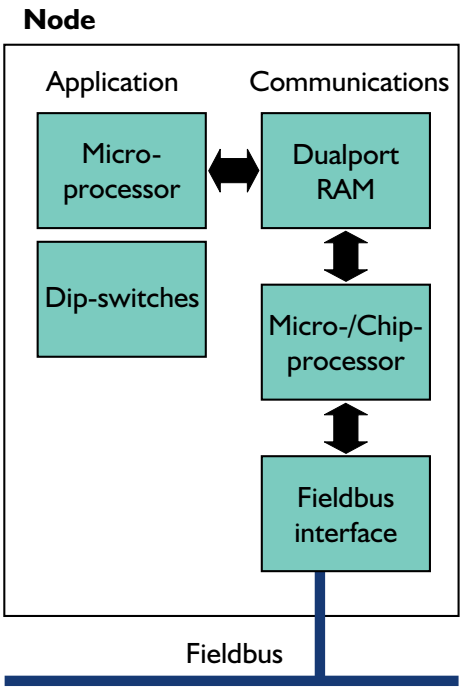


Figure 10: Typicalchip implementation in other fieldbus systems.

P-NET Architecture

P-NET is specified and implemented according to the Open Systems Interconnection Reference Model, on layers 1, 2, 3, 4, and 7, as shown in the diagram in fig. 11.

Normally, a Fieldbus is only implemented on layers 1,2 and 7, but since P-NET features the multi-net structure, the protocol also implements layers 3 and 4.

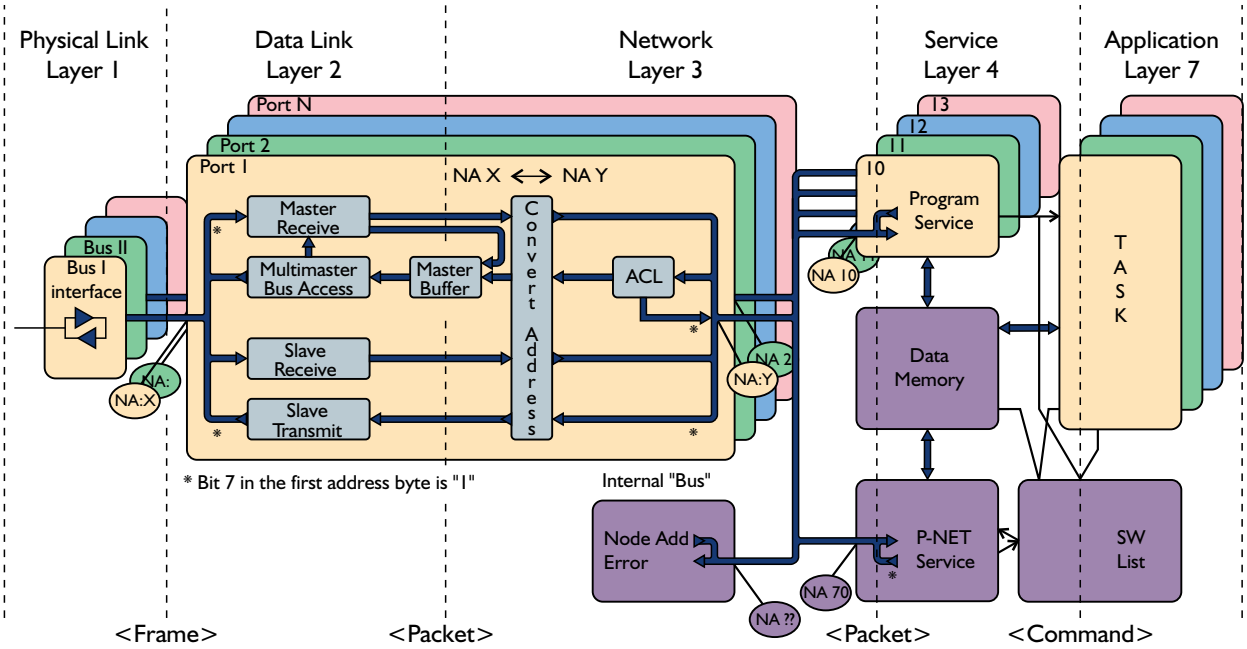


Figure 11: P-NET architecture based on ISO reference model.

Layer 1 is concerned with transmitting raw bits over the bus. It specifies the cable, how a “1” and a “0” is represented on the bus, what the voltage level is, etc.

Layer 2 takes care of the multi-master token, packs the data to be sent into a frame, including source and destination addresses, and performs error detection.

Layer 3 is the P-NET “post office”, which receives and sends the frames according to the destination address. A message may be required to be sent out of another P-NET port, or into the P-NET service, or back to the requesting application, or return a message indicating an unknown address. It also performs the address conversion necessary to ensure a response finds its way back.

Layer 4 handles two different tasks. The first provides the P-NET service, which reads or writes data to internal memory via the SOFTWIRE list, or reroutes a request, if the SOFTWIRE list indicates that the variable is located in another node. The second task holds details about the number ofrequests which have been sent out but are waiting for a reply. When the reply arrives, it is sent back to the calling application task.

Layer 7 is used by application programs to access variables in other nodes. This is done by sending a command block containing references to the SOFTWIRE list, which is where detailed information such as node address, internal address etc. is specified. The SOFTWIRE list is also used for internal variables.

Virtual Token Passing

Each P-NET master is given a node address (NA), between 1 and the number of masters expected within a system.

All masters contain an "*idle bus bit period counter*" which increments for each bit period the bus is idle, but is reset to zero when the bus becomes active. Each master also has an *access counter,* which is incremented when the *idle bus bit period counter* reaches 40, 50, 60, ...

When the *access counter* in a master is equal to its node address, that master holds the token, and is allowed access to the bus. When the access counter exceeds the maximum number of masters, it is preset to 1.

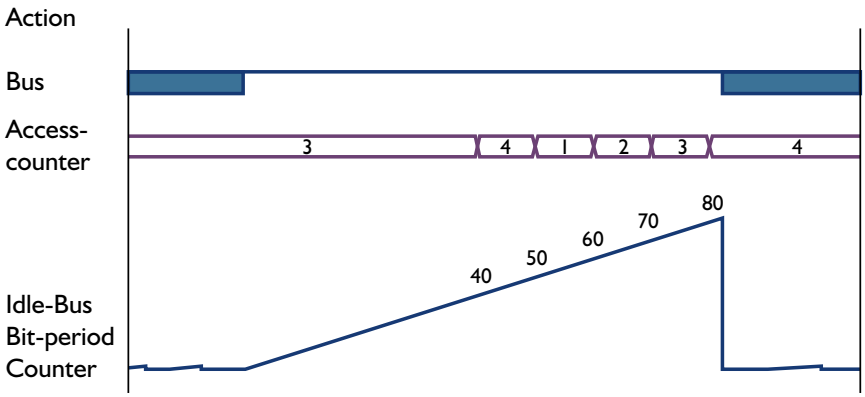


Figure 12: Vitual token passing.

The diagram in fig. 12 shows an example of the token principles in P-NET, within a system configured for 4 masters.

First, master 3 has the token, and is receiving a response from a slave. Then the bus becomes idle.

When 40 idle bit periods have been counted, all access counters are incremented by 1, and master 4 is allowed access to the bus. Since master 4 does not have anything to send, and after 50 bit periods, master 1 is allowed access to the bus.

Master 1 does not need bus use either (it may not even be present), so the virtual token is passed to master 2, when the *idle bus bit period counter* reaches 60.

Since masters 2 and 3 do not require access, the token is eventually passed on to master 4, when the *idle bus bit period counter* is equal to 80. This time, master 4 does require access. Data appears on the bus, so all *idle bus bit period counters* are reset to zero.

The passing of the virtual token takes place within only 130 ìS or 10 bit periods, and no data is actually sent over the bus. A single network can have up to 32 masters with equal priority, and no hierarchy needs to be managed.

Consequently, P-NET does not require any bus arbitrator functions. Virtual token passing is much more efficient than passing the token by message.

P-NET Compared to Dedicated Fieldbus Chip Solutions

Bus systems using special chips (e.g. Profibus FMS), typically receive the complete frame first. Then the chip sends an acknowledge to the master and an interrupt to the host CPU.

Slave processing is then started, and when all data is ready, it is transferred to the chip. Now the Master has to make a second request to obtain the desired result. This is illustrated in the diagram in fig. 13.

P-NET slaves handle the processing of data and the reception and transmission of frames, in parallel. The processing of the request begins in the slave, as soon as the first data bytes arrive. In this way, the standard P-NET data rate of 76,800 bit/s, is not a limiting factor in performance.

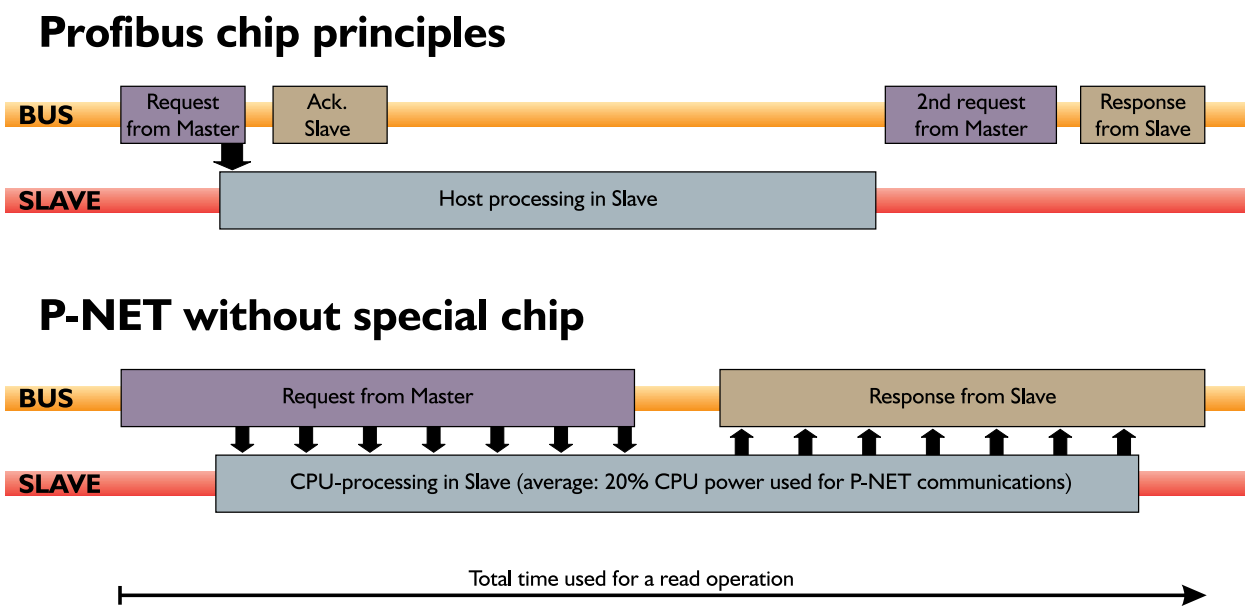


Figure 13: P-NET communication compared to Profibus chip principle.

Every P-NET slave module must answer a request within 390 microseconds (“immediate response”). This eliminates the need for multiple requests for a single variable, or even continuous polling until a result is ready. The immediate response eliminates the need for buffers in the slave to contain a queue of requests or polling from different masters.

The immediate response, coupled with parallel operation and fast token passing, results in a performance similar to other bus systems with a much higher data rate (e.g. 500 kbit/s).

One of the drawbacks of increasing the data rate, is that it leads to a significant reduction in the Fieldbus cable length allowed. For example, at 76.8 kbit/s the bus length can be in the region of 1.2 Km, but at 500 kbit/s the bus length would need to be reduced to 200 m.

The consequence of this is, that for a comparable full size system, 5 extra repeaters would have to be considered for the higher rate system.

International P-NET User Organization

Nearly a 100 companies are now members of the International P-NET User Organization. The membership fee (1995) for companies is 1.500 DKr (approx. 400 DM or £160).

Special arrangements are available for universities and other educational institutions.

By enroling in the International P-NET User Organization, a new member will receive one copy of the P-NET standard.

Members have the right to use the P-NET protocol and the P-NET logo in products, without any royalty.

The International P-NET User Organization arranges International P-NET Conferences, takes part in international Standardization work and disseminates information about P-NET.

Literature references can be requested from the International P-NET User Organization.

To obtain additional information contact the head office or your local society:

Head Office:

- International P-NET User Organization,
  P.O.Box. 192,
  DK-8600 Silkeborg
  Denmark
  Phone +45 87 200 396, Fax +45 87 200 397
  e-mail: P-NET@post4.tele.dk

Local Societies:

- b+ Prof. Dr.-Ing. Jörg Böttcher, Deggendorf, Germany
  Phone +49 991 340 897, Fax +49 991 340 447
  e-mail: 0991340897-0001@t-online.de

- PROCES-DATA (UK) Ltd. , Oxon, England
  Phone +44 (0) 1491 828 200, Fax + 44 (0) 1491 828 201
  e-mail: pnet@easynet.co.uk

- TECNOCON, Vale de Cambra Cedex, Portugal
  Phone +351 56 412 789, Fax +351 56 412 792

- CONFLOW Technologies Inc., Ontario, Canada
  Phone +1 905 840 6800, Fax +1 905 840 6799

Applied P-NET Applications

Factory Environmental Control

Train Fuel Management System at british Rail

Fully Automated Car & Passenger Ferry in Holland

P-NET Conference and Exhibition Participation

Beer Barrel Filling Line in UK Brewery

Milk Truck Data Collection System

Complete Plant for the Concrete Industry

Cheese Production Plant Control System

Fiscal Metering on Oil Distribution Truck