General Purpose Fieldbus Objects

Illustrated by P-NET Channels

Mr. John Johansen
Proces-Data A/S
Navervej 8-10
8600 Silkeborg, Denmark
Phone: +45 87 200 300
Fax: +45 87 200 301
Email: pd@post4.tele.dk

General Purpose Fieldbus Objects

John Johansen

Abstract

P-NET is specified and implemented according to the Open Systems Interconnection Reference Model on layer 1, 2, 3, 4, and 7. Well defined application layer profiles for individual I/O points, gives a well defined Application Interface for programmers.
The I/O interfaces in the different nodes are defined as objects, which contain not only real time data but also predefined function switches, diagnostics, maintenance data and error messages. Within the P-NET Standard, such an object is declared as a **Channel**.
The standardization of the single I/O points instead of complete nodes on the P-NET makes it possible from a master's point of view, to exchange similar I/O's defined within different manufactures' nodes. Nodes can have different I/O structures, but a single I/O will be seen as the same by the master, no matter what kind of node it is part of.

1 General structure of P-NET

P-NET is specified and implemented according to the Open Systems Interconnection Reference Model on layer 1, 2, 3, 4, and 7.

Layer1: The physical link layer is concerned with transmitting raw bits over the network. It describes the electrical interface, baudrate, cabling and so on. RS485, RS 232 and IS 16 are well known electrical interfaces for P-NET.

Layer2: The Data Link Layer controls the Bus Access, create and recognize frame boundaries and node addresses and performs the transmission error control.

Layer3: The Network layer transports packets between layer 2 and 4, and performs the gateway functions.

Layer4: The Service layer holds:
The "Program Service", which receives a Command block from the application layer (7), and
The P-NET service program, which receives a packet from the network layer (3), executes the instruction defined in the packet, and set a status info according to the data, and errors if any.

Layer7: The Application layer is the interface to application programs with well defined data types. The application layer can access variables defined in a structured manner, in the Softwire list.

2 Channels as Fieldbus Objects

An interface module is typically utilized as a signal transmitter for one or more process signals, i.e. a digital output or an analog input. In the following example, a PT100 temperature sensor is mounted in a tank, e.g. a beer storage tank and connected to an interface module, which converts the signal to an understandable and readable value, e.g. in engineering units.

Each process signal involves more information than only the state or the value of the signal. Other various parameters, variables, are related to the process signal as well as specific functions for converting, scaling, filtering etc. All the calculations, which are predefined in the functions, are performed in the interface module.

This collection of related variables and functions for a single process signal is a Process Object, and it is called a Channel.

These variables, including the process value, are found on fixed logical addresses, called SoftWire numbers. The data type is also fixed for these variables.

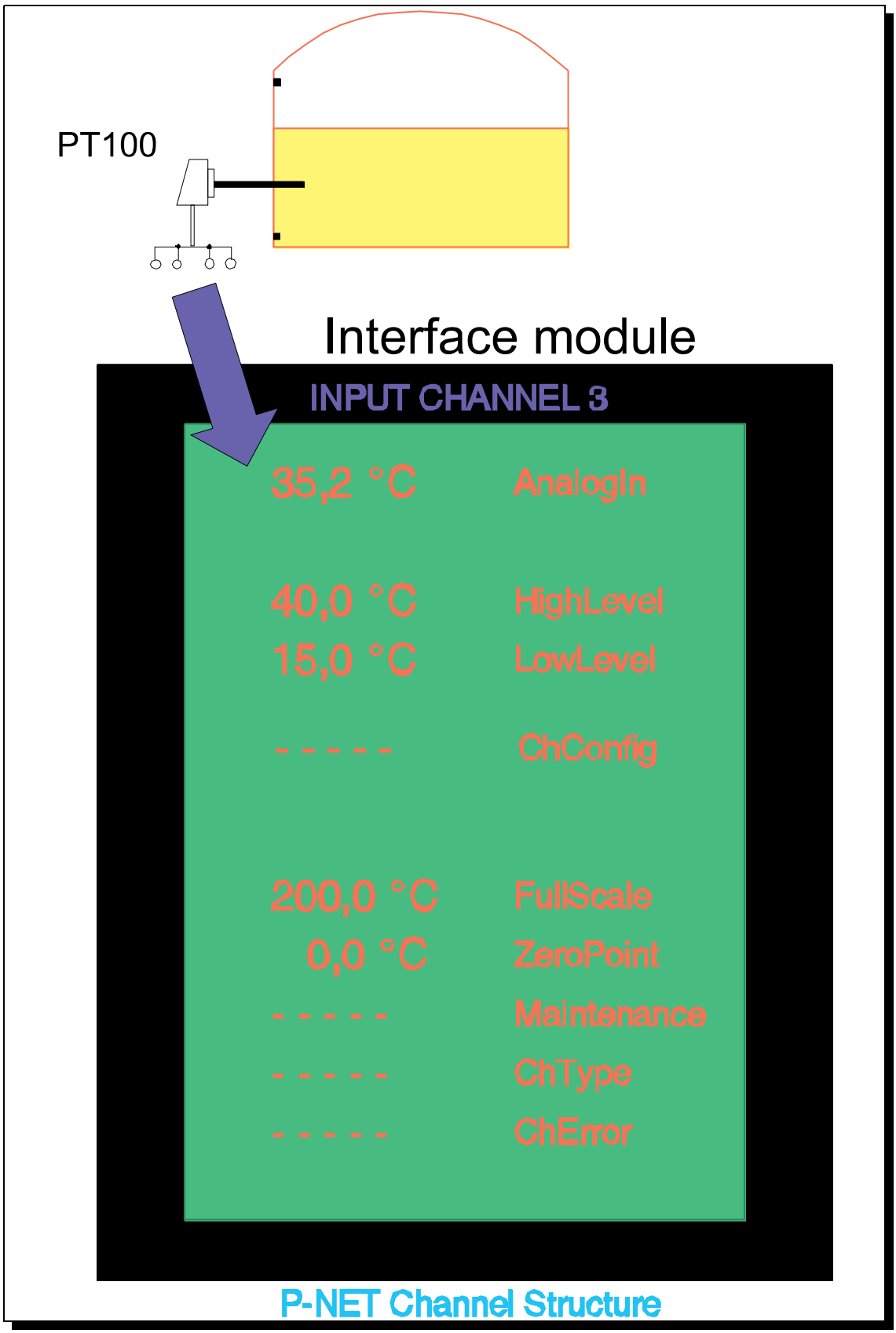
One of the important features of P-NET is error message handling. Therefore, each Channel has an Error Code, ChError. This Error Code contains error information related to the Channel only. The Error Code is always located at SWNo xxxF, and is of a fixed type. Examples of errors: Overload, Signal disconnected etc.

Each channel in an interface module is described in an individual ChType variable. This is a Record, consisting of a unique number for the channel type, and a TRUE boolean value for each of the registers which are represented within a channel, The register number in a channel, corresponds to the index number in the boolean array. Depending on the channel type, more fields can be found in this record.

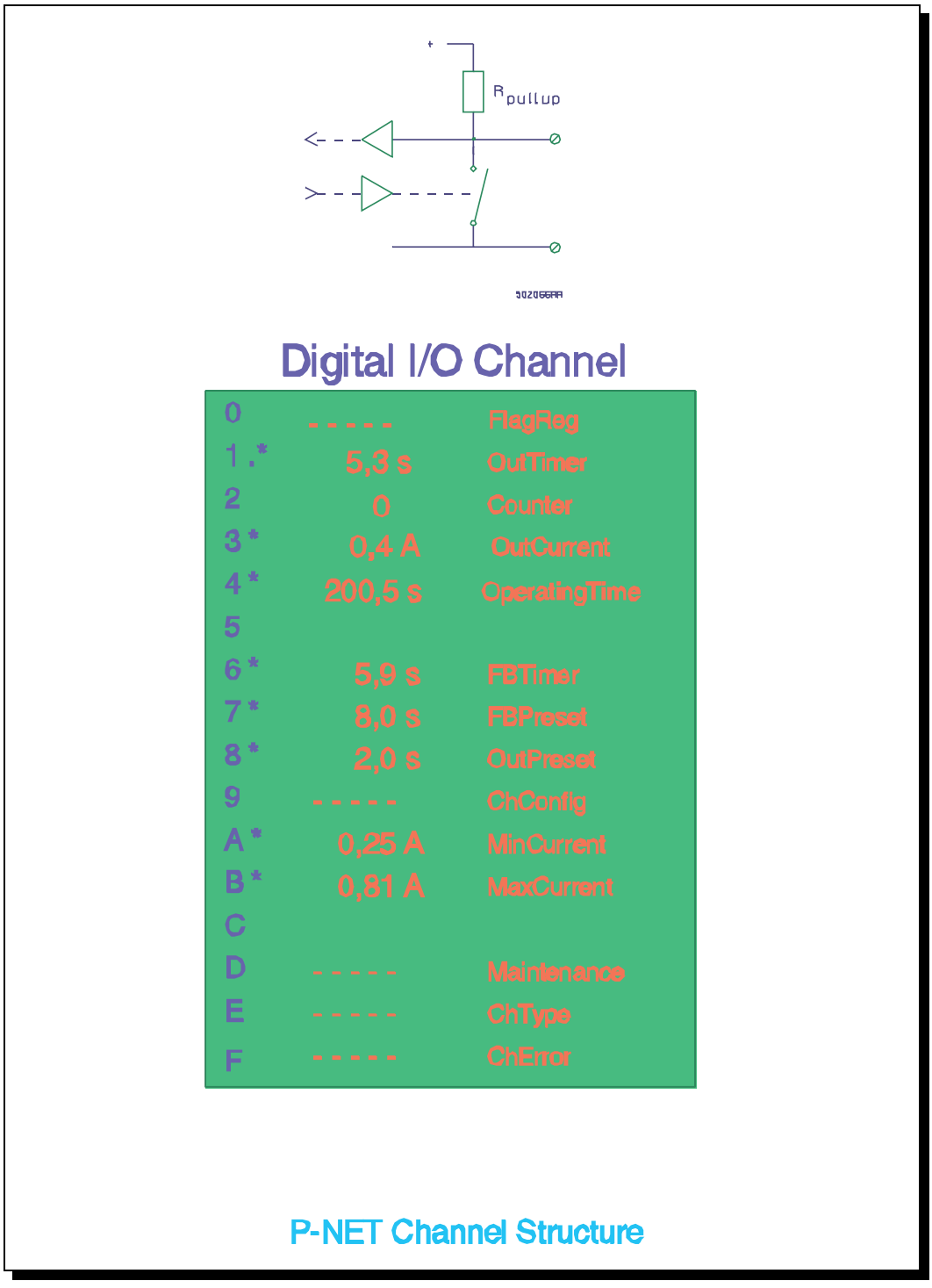
Additional process related information can be connected to the process signal, such as HighLevel and LowLevel, which can be used as internal "limitswitches".
A variable for storing Maintenance information for the process signal in question is also found, e.g. date and type for the latest service inspection.

A Channel can hold up to 16 registers, each having their own SoftWire number (SWNo). These 16 variables or constants within a Channel can be of any type, including complex, and located in different memory technologies (ROM, RAM, EEPROM....). Some of the SWNo's can be declared as unused.

The P-NET standard for communication concerns only one variable at a time and a Channel holds a number of independent variables. Therefore a complete Channel cannot be transferred in a single transmission.


PT100
Interface module
INPUT CHANNEL 3
35,2 °C
AnalogIn
40,0 °C
HighLevel
15,0 °C
LowLevel
- - - - -
ChConfig
200,0 °C
FullScale
0,0 °C
ZeroPoint
- - - - -
Maintenance
- - - - -
ChType
- - - - -
ChError
P-NET Channel Structure

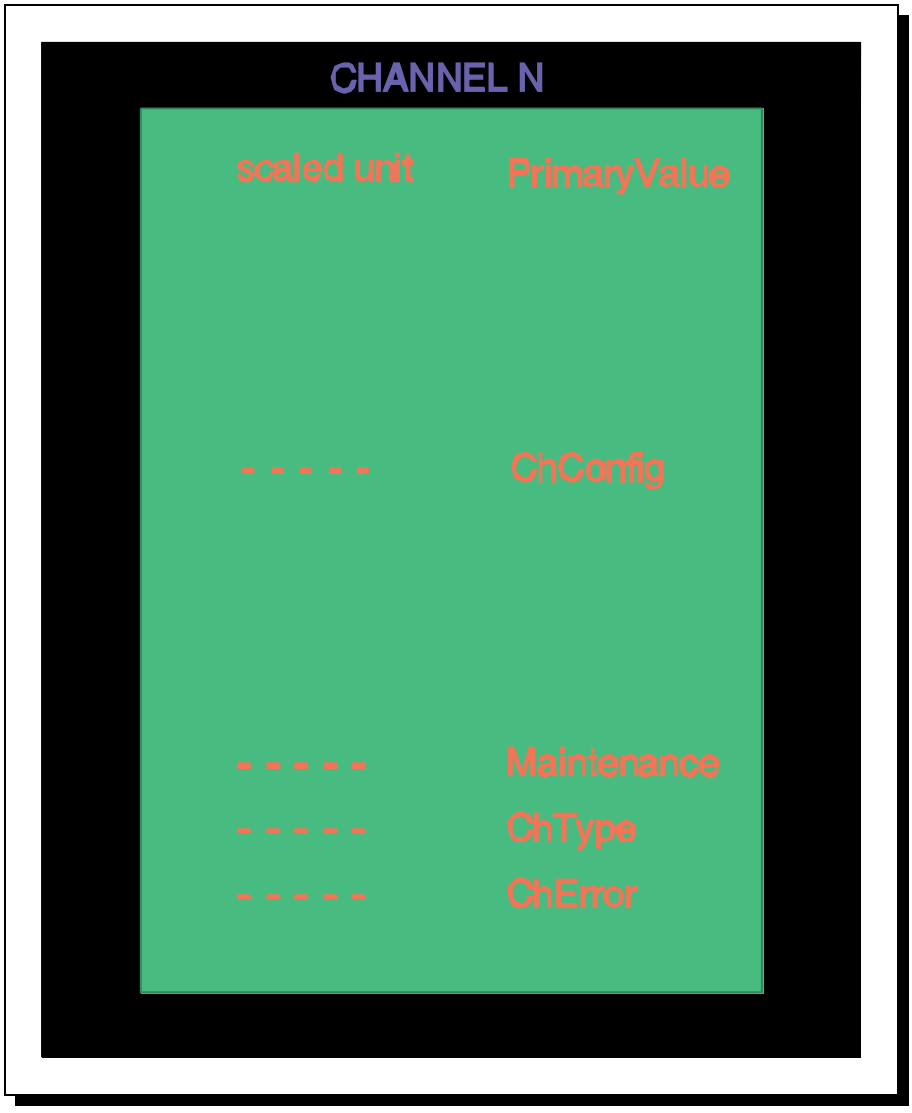
Another example of an interface Channel is a digital output, where the state of the output can be ON or OFF. The output in an interface module is controlled via a microprocessor, so a counter for counting output activations is very easily added as well as a register for summarizing the operating time. The current in the load on the output could also be measured, and there may be various automatic functions to select for the output. A register must also be found to indicate errors, the ErrorCode register. All this information concerning one single output can be gathered in a Channel.



Not only process signals, but also other kinds of signals, and the corresponding parameters, can be organized as channels. Such a Channel could be a result of a calculation, such as regulator signals, i.e. a PID regulator, a printer channel, a barcode reader channel, etc.

The standardization of the single I/O points instead of complete nodes on the P-NET makes it possible from a master's point of view, to exchange similar I/O's defined within different manufactures' nodes as long as the channel type is the same.

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



Associated with the Application layer within P-NET, is the data format standardization, which includes reals, bytes, strings, boolean, but also more complex entities, as arrays and records. All measurement values sent over P-NET are already scaled in engineering units.

3 Common principles of interface modules

The modules are constructed according to some common principles, which are mentioned in the following:

1. The measured values are represented as a scaled and calibrated real number in engineering units, as the modules calculate the measured values from input signals and calibrating constants stored in non volatile memory e.g. battery RAM or EEPROM. This method has the advantage, that the various masters being connected need not know all calibrating constants, and in case of exchange of a module, the masters need not to be adjusted.

2. The modules have an internal automatic test system, supervising the following internal functions: micro-processor, ROM, RAM, EEPROM, A/D converter, and parameters with specified limits. In case of an error on a channel, all response frames from the module concerning that channel, will contain a status bit indicating an error in the channel. Moreover, the module stores the character of the error in an error status register on the channel, which the master can read.

3. When calling an interface module via the P-NET, "logical addressing", a SoftWire number, is normally used. This means that when the module gets an address via the P-NET, a physical address is found via a table, the SoftWire table. This principle makes it possible to reorganize the module, without this having any affect on the SoftWire numbers.

4. All modules have a common channel 0, named Service Channel. This channel contains functions and registers common to the entire module.

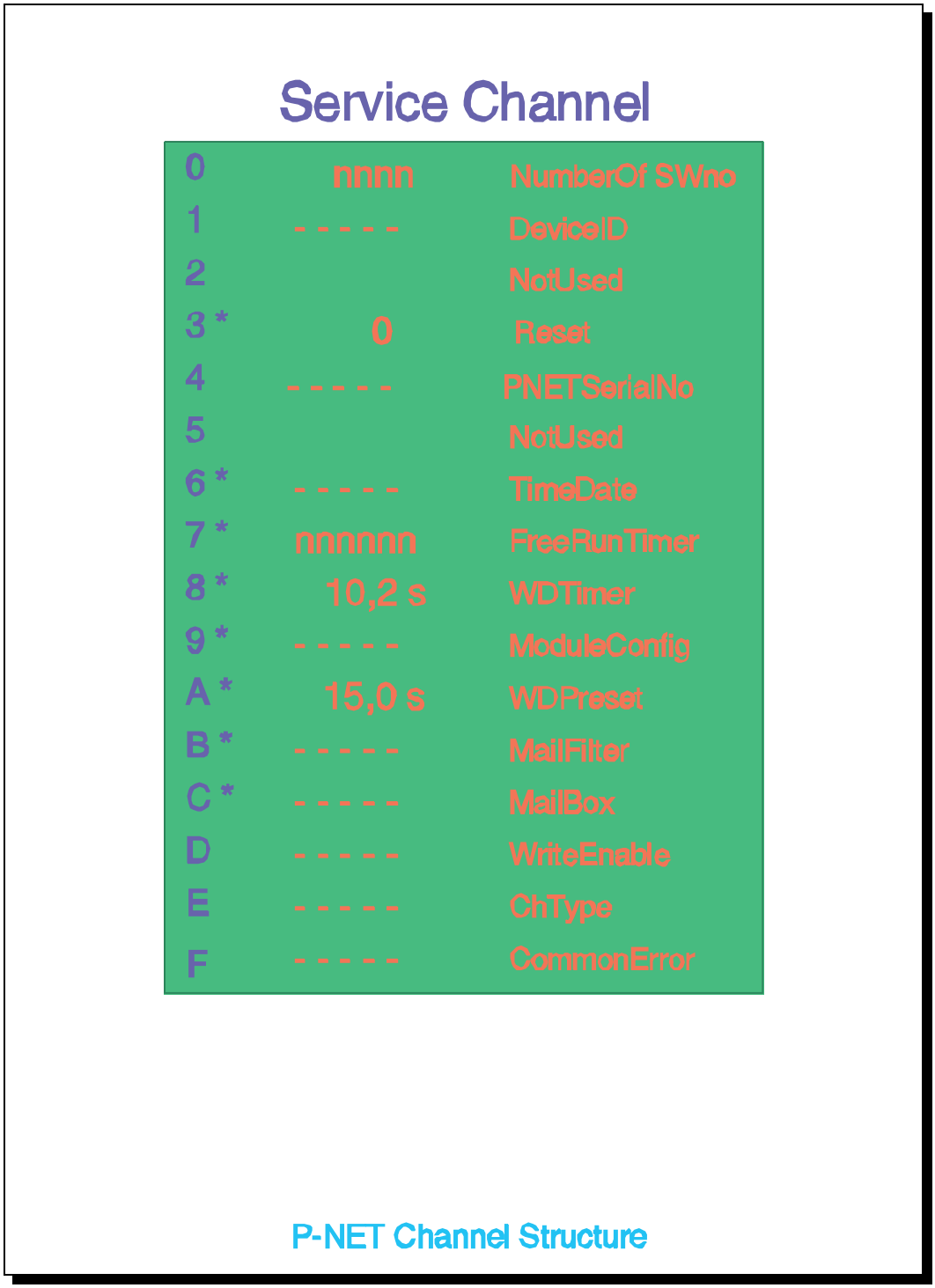
5. All variables on SWNo's must be available for immediate access via P-NET. If there are any problems in instantly retrieving the value for some of the variables (e.g. stored in EEPROM), then the status bit in the P-NET communication should be used (busy or wait), until the data are fetched. If the variable is of a complex type, all the fields or elements should be located in the right order, as a contiguous collection of bytes. This will simplify the communication program when the entire variable is accessed.

6. In a complex variable, all fields or elements must be stored in the same memory type.

7. A channel should only concern data available for the user. If any data must be available for the manufacturer only, these data should be accessed via abs. addresses, or at a "hidden", non standard channel. This also concerns special functions, e.g. accessing EEPROM for setting the serialnumber in the device.

The service channel includes the first 16 SWNo with fixed types. This Service Channel makes it possible to create universal programs for identification of an "unknown" device. (SWNo 1).

The Service channel must always be present in a P-NET interface module.



P-NET Channel Structure