1. PROFINET Basis

PROFINET is the automation standard of PROFIBUS&PROFINET International (PI). It is 100% Ethernet-compatible as defined in IEEE Standards. Several specifications exist, which describe different system aspects:

PROFINET IO is used for data exchange between I/O controllers (PLC, etc.) and I/O devices (field devices). PROFINET IO uses the proven communication model and application view of PROFIBUS DP and extends it by Ethernet as the communication medium. Among other benefits, this provides a greater bandwidth and allows more stations on the network. The PROFINET IO specifications define a protocol and an application interface for exchanging I/O data, alarms and diagnostics and for transmitting data records and logbook information. To exchange I/O data and alarms, PROFINET IO is based directly on the Ethernet protocol. This real-time (RT) solution allows response times in the range of 5 ms, which corresponds to today's PROFIBUS DP applications. If it has to be even faster and if data exchange should be performed isochronously (IRT), a special chip is used, which also supports switch functions. "Normal" Ethernet communication is of course also possible when using the chip. The solution consists in reserving bandwidth for the isochronous data exchange and bandwidth for "the remainder." Innovations have also been made with regard to the device description in that XML is used for structuring the information.

For all that is new, however, the existing has not been forgotten or dispensed with. The integration of existing fieldbus devices will be performed via proxies, and PROFIBUS profiles will also be available for PROFINET IO. PROFIdrive and PROFIsafe will be the first to be revised.

PROFINET CBA (Component based Automation) is suitable for component- based machine-to-machine communication via TCP/IP and for real-time communication to meet realtime requirements in modular plant manufacturing. TCP/IP or Ethernet is used as the transport protocol. The response times range around 100 ms (nondeterministic) or 5 - 10 ms (deterministic, RT solution).

Besides the PROFINET IO view of data, PROFINET CBA also supports a component view. In this view, the application software and the devices are modeled as self-contained components which interact via a component bus. It enables a simple modular design of plants and production lines based on distributed intelligence using graphics-based configuration of communication between intelligent modules (PROFINET CBA Engineering).

PROFINET CBA and **PROFINET IO** can be operated separately and in combination such that a **PROFINET IO** unit appears in the plant view as a **PROFINET CBA plant**.

PROFINET CBA and **PROFINET IO** devices can be used together in one application and both protocol variants can be implemented in the same device.

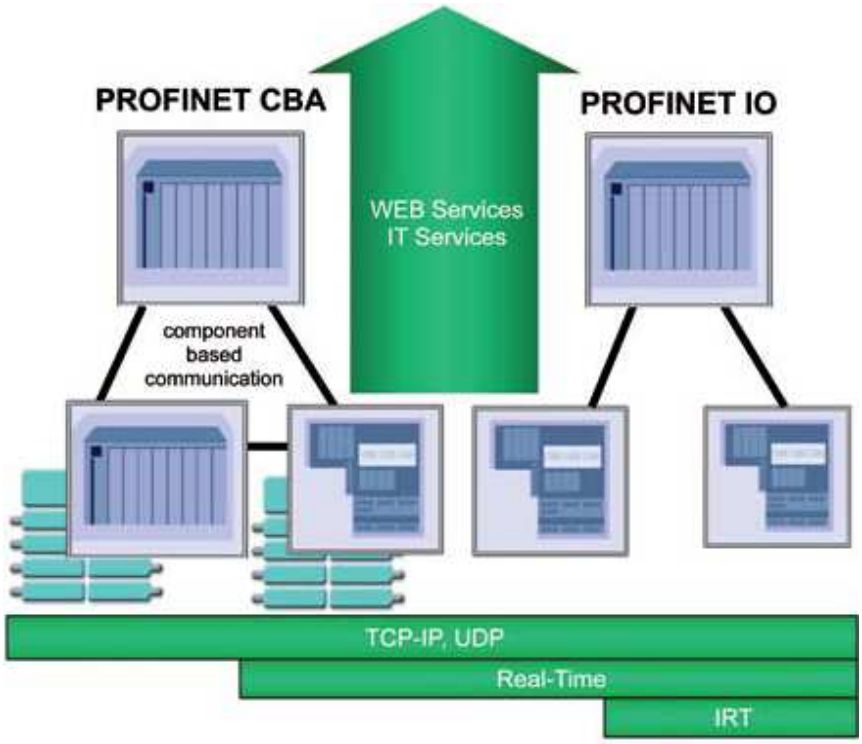

Fig. 1.1 PROFINET Basis

Further Specifications The PROFINET WebIntegration Specification defines the possibilities of using web technologies for access to PROFINET data. The PROFINET Network Management Specification describes how standards from the area of network management are applied to the PROFINET environment. The use of IT protocols also requires considering the corresponding security aspects. The PROFINET Security Specification will define a framework for assessing the risk potential and the appropriate solutions.

2. PROFINET IO

1.1 Profinet IO device clasification

PROFINET IO classifies devices into three types. **IO-Controllers** are devices that execute an automation program. Controllers, functionally similar to a PROFIBUS Class 1 Master, exchange data with IO-Devices. **IO-Devices** are distributed sensor/actuator devices connected to the IO-Controller over Ethernet. In PROFIBUS terms, IO-Devices are similar to PROFIBUS slaves. **IO-Supervisors** are HMIs, PCs or other commissioning, monitoring, or diagnostic analysis devices. These devices are similar to Class 2 PROFIBUS Masters.

IO-Controllers map IO data from PROFINET IO devices into the process image of the controller. In Siemens S7 Programmable Controllers, IO data, alarms, and status data are mapped into the process image in much the same way it is done for PROFIBUS devices. These data values are then available for use by the control program. IO-Controllers must support the following kinds of services:

- **Cyclic Data Exchange** – The exchange of data between IO-Controllers and IO-Devices.
- **Acyclic Data Exchange** – The exchange of Configuration and Diagnostic data
- **Alarms** – Alarm data exchange from an IO-Device to an IO-Controller
- **Context Management** – Connection processing

IO-Supervisors are used for commissioning and diagnostic data collection. IO-Supervisors can read and write internal diagnostic data associated with the PROFINET IO stack or diagnostic data provided by the application program of a device. IOSupervisors can also read and write configuration data using special, non-cyclic record data object services. These types of devices may only be used during the commissioning process or they may be used as an HMI to display diagnostic data to the end user.

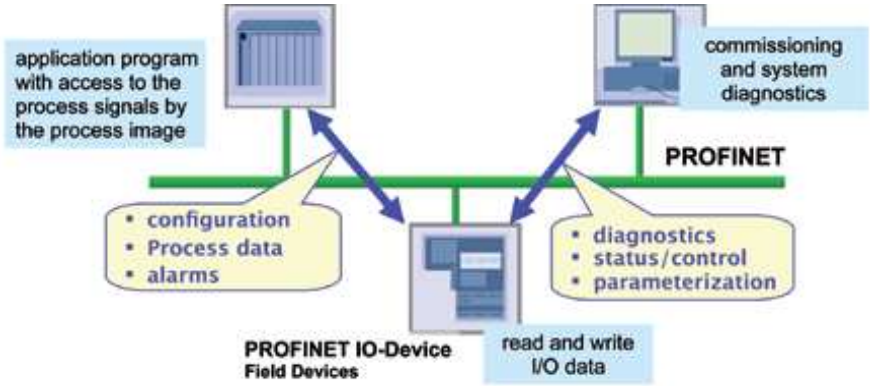

Fig. 2.2 Profinet IO device cpecification

A PROFINET IO system requires at least one IO-Controller and one IO-Device. Systems can be configured in various configurations; multiple IO-Controllers for a single IO-Device; single IO-Controllers for multiple IO-devices; and multiple IOControllers with multiple IO-Devices.

1.2 PROFINET IO device addressing

Bit value 47 ... 24						Bit value 23 ... 0					
0	0	0	E	C	F	X	X	X	X	X	X
Company code → OUI						Consecutive number					

Each PROFINET device is addressed based on a MAC address. This address is unique worldwide. The company code (bits 47 to 24) can be obtained from the IEEE Standards Department free of charge. This part is called the OUI (organizationally unique identifier).

Profibus International (PI) offers MAC addresses to device manufacturers that do not want to apply for their own OUI, in other words, a defined OUI and the manufacturerspecific portion (bits 23 to 0). This service allows components to acquire MAC addresses from the PI Support Center. The assignment can be completed in 4 K-ranges.

The OUI of PI is 00-0E-CF and is structured as shown in the table. The OUI can be used for up to 16,777,214 products.

PROFINET IO field devices are addressed using MAC addresses and IP addresses. Figure 2.2 shows a network that comprises two subnets. These are represented by the different network_IDs (subnet mask). For PROFINET IO field devices, address resolution is based on the symbolic name of the device, to which a unique MAC address is assigned. After the system is configured, the engineering tool loads all information required for data exchange to the IOController, including the **IP addresses** of the connected IO-Devices. Based on the name (and the associated MAC address), an IO-Controller can recognize the configured field devices and assign them the specified IP addresses using the DCP protocol (Discovery and Configuration Protocol) integrated in PROFINET IO. Alternatively, addressing can be performed via a DHCP serve Following address resolution, the system powers up and parameters are transmitted to the IO-Devices. The system is then available for productive data traffic.

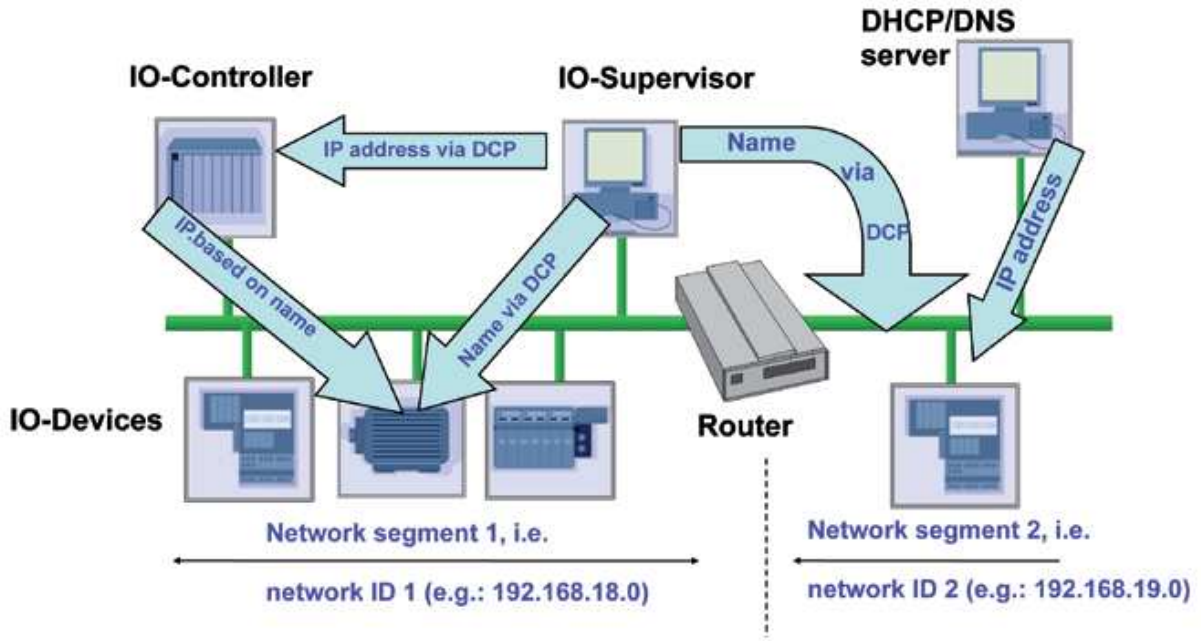

Fig. 2.3 Profinet IO addressing.

PROFINET IO field devices are always connected via switches as network components. This takes the form of a star topology with separate multiport switches or a line topology with switches integrated in the field device (2 ports occupied). Within a network, a PROFINET IO field device is addressed by its device MAC address.

PROFINET transmits some message frames (e.g., for synchronization, neighborhood detection) with the MAC address for the respective port and not the device MAC address. For this reason, each switch port in a field device requires a separate port MAC address. Therefore, a 2-port field device has 3 MAC addresses in the as-delivered condition. However, these port MAC addresses are not visible to users. Because the field devices are connected via switches, PROFINET always sees only point-to-point connections (same as Ethernet). That is, if the connection between two field devices in a line is interrupted, the field devices located after the interruption are no longer accessible. If increased availability is required, provision must be made for redundant communication paths when planning the system, and field devices/switches that support the redundancy concept of PROFINET must be used. PROFINET-suitable switches

must support “Autonegotiation“ (negotiating of transmission parameters) and “Autocrossover“ (crossing of send and receive lines in the switch). As a result, communication can be established autonomously, and fabrication of the transmission cable is uniform. Only cables wired 1:1 are used.

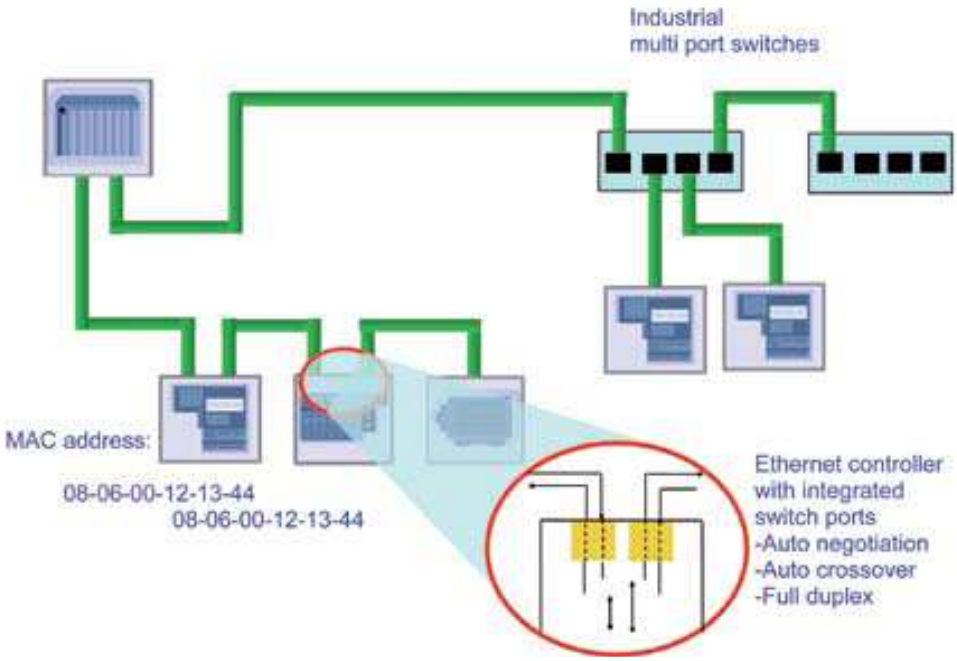

Fig. 2.4 PROFINET IO field devices are always connected via switches

1.3 PROFINET IO device model

To facilitate understanding of process data addressing in a PROFINET IO field device, an overview of the device modeling and, thus, addressing of I/O data in an automation system is advantageous. For field devices, a distinction is made between:

- Compact field devices (the degree of expansion is defined in the as-delivered condition and cannot be changed to meet future requirements).
- Modular field devices (for different applications, the degree of expansion can be customized to the use case when configuring the system).

fig. 2.5 Compact and modular field device

All field devices are described in terms of their available technical and functional properties in a GSD file (General Station Description) to be created by the field device developer. It contains among other things a representation of the device model that is

reproduced by the DAP (Device Access Point) and the defined modules for a particular device family. A DAP is, so to speak, the bus interface (access point for communication) to the Ethernet interface and the processing program.

It is defined along with its properties and available options in the GSD file. A variety of I/O modules can be assigned to it in order to manage the actual process data traffic.

The proven device model of PROFIBUS has been largely applied for PROFINET IO and adapted to the requirements of the end user. This results in an additional nesting depth (slot and subslot) for PROFINET IO. **The following addressing options are standardized:**

The **slot** designates the physical slot of an I/O module in a modular I/O field device in which a module described in the GSD file is placed. The configured modules containing one or more subslots (actual I/O data) for data exchange are addressed on the basis of the different slots.

Within a slot, the **subslots** form the actual interface to the process (inputs/outputs). The granularity of a subslot (bitwise, bytewise, or wordwise division of I/O data) is determined by the manufacturer. The data content of a subslot is always accompanied by status information, from which the validity of the data can be derived.

The **index** specifies the data within a slot/subslot that can be read or written acyclically via read/write services. For example, parameters can be written to a module or manufacturer-specific module data can be read out on the basis of an index.

The device model is shown below for a modular IO-Device configuration (bus interface and three input/output modules).

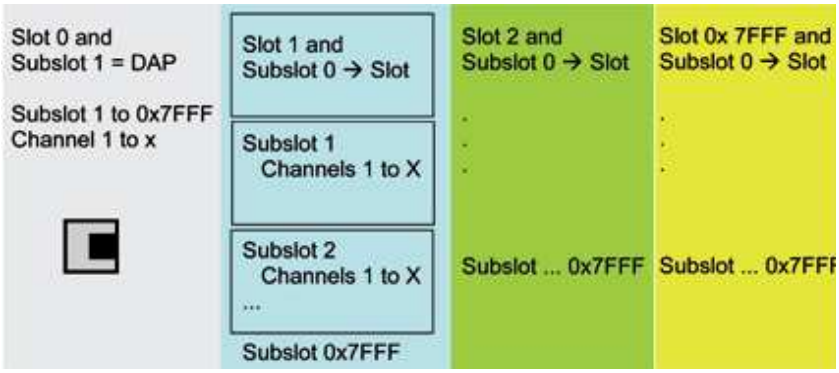

Fig. 2.6 Addressing of IO data (Slot and Subslots)

The manufacturer uses definitions in the GSD file to describe the number of slots/subslots an IO-Device can process.

Cyclic I/O data are addressed by specifying the slot/subslot combination. These can be freely defined by the manufacturer.

For acyclic data traffic via read/write services, an application can specify the exact data to be addressed using slot, subslot. For demand-oriented data exchange, the third addressing level, i.e., the index is added. The index defines the function that is to be initiated via the slot/subslot combination (e.g., reading of input data of a subslot, reading of I&M functions, reading of actual/desired configuration, etc.).

To prevent the possibility of competing accesses in the definition of user profiles (e.g., for PROFIdrive, weighing and dosing, etc.), it is appropriate to define not only slots and subslots but also an additional addressing level, i.e., the API (**A**pplication **P**rocess **I**dentifier/Instance). This

degree of freedom enables different applications to be handled separately in order to prevent overlapping of data areas (slots and subslots).

2. Communication in PROFINET IO

PROFINET IO provides protocol definitions for the following services:

- Address resolution for field devices
- Cyclic transmission of I/O data (RT and IRT)
- Acyclic transmission of alarms to be acknowledged
- Acyclic transmission of data (parameters, detailed diagnostics, I&M data, information functions, etc.) on an as needed basis.
- *Redundancy mode for realtime frames*

The combination of these communication services in the higher-level controller makes it possible to implement convenient system diagnostics, topology detection, and device replacement, among other things.

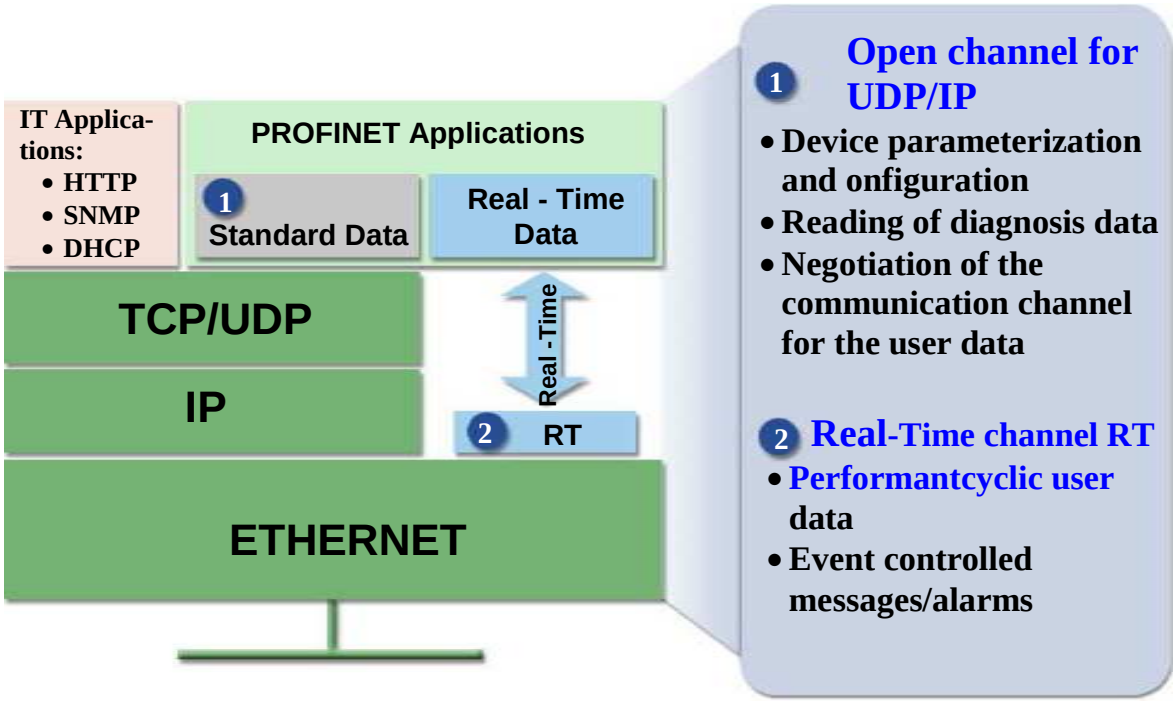

Fig. 3.1

2.1 Principles of real-time communication in PROFINET IO

Standard Ethernet communication via TCP(UDP)/IP communication is sufficient for data communication in some cases. In industrial automation, however, requirements regarding time behavior and isochronous operation exist that cannot be fully satisfied using the UDP/IP channel.

A scalable real-time concept is the solution for this. With RT, this concept can be realized with standard network components, such as switches and standard Ethernet controllers. RT communication takes place without TCP/IP information.

The transmission of RT data is based on cyclical data exchange using a provider/consumer model. The communication mechanisms of layer 2 (according to the ISO/OSI model) are sufficient for this. For optimal processing of RT frames within an IO-Device, the VLAN tag according to IEEE 802.1Q (prioritization of data frames) has been supplemented with a special Ethertype that enables fast channelization of these PROFINET frames in the higher-level software of the field device.

Ethertypes are allocated by IEEE and are therefore an unambiguous criterion for differentiation among Ethernet protocols. Ethertype 0x8892 is speci fied in IEEE and is used for fast data exchange in PROFINET IO.

To enable enhanced scaling of communication options and, thus, also of determinism in PROFINET IO, real-time classes have been defined for data exchange. From the user perspective, these classes involve unsynchronized and synchronized communication. The details are managed by the field devices themselves.

Real-time frames are automatically prioritized in PROFINET compared to UDP/IP frames. This is necessary in order to prioritize the transmission of data in switches to prevent RT frames from being delayed by UDP/IP frames. PROFINET IO differentiates the following classes for RT communication. They differ not in terms of performance but in determinism.

RT_CLASS_1: Unsynchronized RT communication **within a subnet**. No special addressing information is required for this communication. The destination node is identified using the ‚Dest. Addr.' only. Unsynchronized RT communication within a subnet is the usual data transmission method in PROFINET IO. If the RT data traffic has been restricted to one subnet (same network ID), this variant is the simplest. This communication path is standardized in parallel to UDP/IP communication and implemented in each PROFINET IO field device. A deliberate decision was made here to eliminate the management information of UDP/IP and RPC. The RT frames received are already identified upon receipt using the Ethertype (0x8892) and forwarded to the RT channel for immediate processing. Industrial standard switches can be used in this RT class.

RT_CLASS_2: frames can be transmitted via synchronized or unsynchronized communication. Unsynchronized communication in this case can be viewed exactly the same as RT_CLASS_1 communication.

In synchronized communication, the start of a bus cycle is defined for all nodes. This specifies exactly the allowable time base for field device transmission. For all field devices participating in RT_CLASS_2 communication, this is always the start of the bus cycle. PROFINET-suitable switches must support this synchronization for this communication class. This type of data transmission, which has been designed for performance, brings with it specific hardware requirements (Ethernet controller/switch with support of isochronous operation).

RT_CLASS_3: Synchronized communication within a subnet. During synchronized RT_CLASS_3 communication, process data are transmitted with maximum precision in an exact order specified during system engineering (maximum allowable deviation from start of bus cycle of 1 µs). With the aid of topology-optimized data transmission, this is also referred to as IRT functionality (Isochronous Real-Time). In RT_CLASS_3 communication, there are **no wait times**. In order to take advantage of the data transmission designed for maximum performance, special hardware requirements apply (Ethernet controller with support of isochronous operation).

RT_CLASS_UDP: The unsynchronized cross-subnet communication between different subnets requires addressing information via the destination network (IP address). This variant is also referred to as RT_CLASS_UDP. Standard switches can be used in this RT class.

For RT frames, data cycles of 5 ms at 100 Mbps in full duplex mode with VLAN tag are sufficient. This RT communication can be realized with all available standard network components.

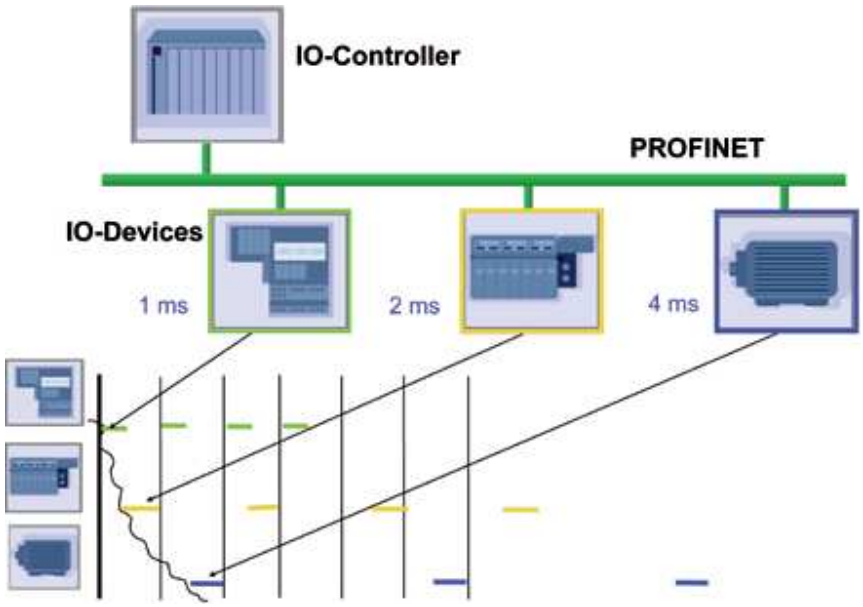

Fig. 3.2

2.2 Cyclic data traffic

Cyclic I/O data are transmitted unacknowledged as real-time data between provider and consumer in a parameterizable resolution. They are organized into individual I/O elements (subslots). The connection is monitored using a watchdog (time monitoring mechanism). During data transmission in the frame, the data of a subslot are followed by a provider status. This status information is evaluated by the respective consumer of the I/O data. It can use this information to evaluate the validity of the data from the cyclic data exchange alone. In addition, the consumer statuses for the counter direction are transmitted. Diagnostics are no longer directly required for this purpose. For each message frame, the ‚Data Unit' (trailer) is followed by accompanying information regarding the global validity of data, redundancy, and the diagnostic status evaluation (data status, transfer status). The cycle information (cycle counter) of the provider is also specified so that its update rate can be determined easily. Failure of cyclic data to arrive is monitored by the respective consumer in the communication relation. If the configured data fail to arrive within the monitoring time, the consumer sends an error message to the application.

2.2.1 Acyclic data traffic

Acyclic data exchange can be used to parameterize and configure IO-Devices or to read out status information. This is accomplished with read/write frames via standard IT services using

UDP/IP. In addition to the data records available for use by device manufacturers, the following system data records are also specially defined:

- **Diagnostic information** can be read out by the user from any device at any time.
- **Error log entries** (alarms and error messages), which can be used to determine detailed timing information about events within an IO-Device.
- **Identification information** as specified in PNO Guideline „I&M Functions".
- **Information functions** regarding real and logical module structuring.
- **Readback of I/O data.**

An index is used to distinguish which service is to be executed with the read/ write services.

2.2.2 Multicast Communication Relation (MCR)

For data exchange with multiple parameters, Multicast Communication Relation (MCR) has been defined. This allows direct data traffic from a provider to multiple nodes (up to all nodes) as direct data exchange. MCRs within a segment are exchanged as RT frames. Cross-segment MCR data follow the data exchange of the RT class. ‚RT_CLASS_UDP'. Data that are exchanged via MCR are subject to the IO-Device model and are assigned to subslots. An M-provider subslot of an IO-Device can publish the input data both to the assigned IO-Controller via an input CR and via a multicast communication relation (M-CR). Different transmission methods (RT, IRT) can be used for the two CRs.

2.2.3 Event-oriented data traffic

In PROFINET IO, the transmission of events is modelled as part of the alarm concept. These include both systemdefined events (such as removal and insertion of modules) and user-defined events detected in the control systems used (e.g., defective load voltage) or occurring in the process being controlled (e.g., temperature too high). When an event occurs, sufficient communication memory must be available for data transmission to ensure against data loss and to allow the alarm message to be passed quickly from the IO-Device. The application in the data source is responsible for this. Alarms are included in acyclic RT data.

3. Diagnostics Concept of PROFINET IO

PROFINET IO transmits high-priority events mainly as alarms. These include both system-defined events (such as removal and insertion of modules) and user-defined events (e.g., defective load voltage) detected in the control systems used or occurring in the process (e.g., boiler pressure too high).

Diagnostic and status messages represent another means of forwarding information regarding incorrect behavior in a system. These are not transmitted actively to the higher-lever controller. In order to assign them explicitly, PROFINET distinguishes between process and diagnostic alarms.

Process alarms must be used if the message originates from the connected process, e.g., a limit temperature was exceeded. In this case, the IO-Device may still be operable. The data are not saved locally in the submodule.

Diagnostic alarms must be used if the error or event occurs within an IO-Device (or in conjunction with the connected components, such as a wire break). Diagnostic and process

alarms can be prioritized differently by the user. In contrast to process alarms, diagnostic alarms are identified as incoming or outgoing.
The overview figure 4.1 shows the main structure of an alarm message or diagnostic entry used to signal an event. Any fatal error is always signaled as an alarm. Every alarm triggers an entry in the diagnostic buffer.

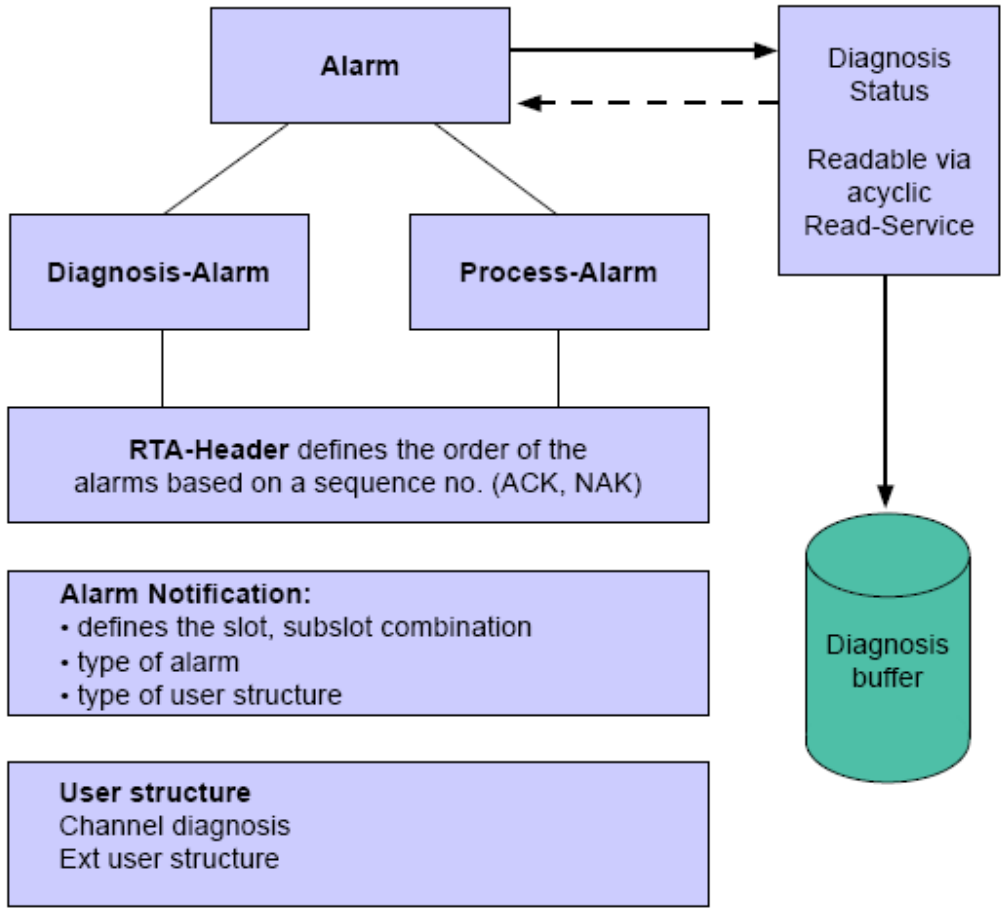

Fig. 4.1 Structure of an alarm message

The network diagnostics is part of the diagnostics management and contributes significantly to the reliability of the network operation. For maintenance purposes and for monitoring the network components, SNMP (Simple Network Management Protocol) has been established as the international standard. SNMP allows both read access and write access (for administration) of network components in order to read out statistical data pertaining to the network as well as port-specific data and information regarding the neighborhood detection. Integration of monitoring functions directly into network components such as switches represents another diagnostic means for networks. IEEE-compliant standard switches are designed exclusively for the purpose of forwarding diagnostic information of the connected field devices to an IO-Controller. Additional monitoring functions are not usually integrated. Switches designed as IODevices feature a higher level of integrated intelligence.

For system power-up, the IO-Controller transfers the connect frame containing the ‚CMInactivityTimeout-Factor', which is used to monitor the system power-up. This monitoring time ends after the first valid data exchange between IO-Controller and IO-Device and is then replaced by the watchdog function.
In PROFINET IO communication, the cyclic data traffic between provider and consumer is monitored by the watchdog function, which is integrated by default. Cyclic data including status information are transmitted between the IO-Controller and IO-Device. A consumer detects failure of the communication connection based on expiration of the watchdog. The application in the consumer is thereby informed. The response to this must be defined on a user-specific basis.

3.1 Mode of Operation of PROFINET IO

System engineering and GSD To enable system engineering, the GSD files (General Station Description) of the field devices to be configured are required. The field device manufacturer is responsible for supplying these. During system engineering, the configuring engineer joins together the modules/submodules defined in the GSD file to map them to the real system and to assign them to slots/subslots. The configuring engineer configures the real system, so to speak, symbolically in the engineering tool.

Device identification though name assignment A logical name is assigned to every field device. It should reference the function or the installation location of the device in the plant and ultimately lead to assignment of an IP address during address resolution. The name can always be assigned with the DCP protocol (Discovery and Configuration Protocol) integrated by default in every PROFINET IO field device. Because DHCP (Dynamic Host Configuration Protocol) has found widespread use worldwide and, for example, address setting via DCP requires additional effort for MS Windows-based IO field devices, PROFINET provides the option for address setting via DHCP or other The addressing options supported by an PROFINET IO field device are defined in the GSD file for the respective device.

Each IO-Controller manufacturer also provides an engineering tool for configuring a plant. Figure 4.2 shows the relationship between GSD definitions, the configuration, and the real plant view.

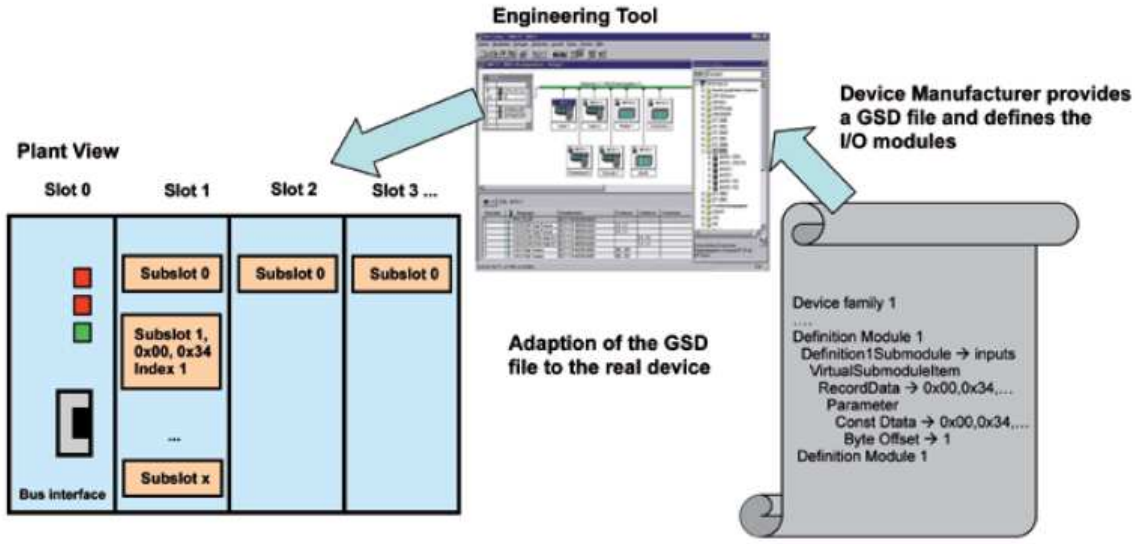

Fig. 4.2 Relationship between GSD definitions, the configuration, and the real plant view

Download of plant information After completion of the plant engineering, the configuring engineer downloads the plant data to the IOController, which also contains the plant-specific application. As a result, an IO-Controller has all the information needed for addressing the IO-Devices and for data exchange.

Address resolution Before it can perform data exchange with an IO-Device, the IO-Controller must assign the IO-Device an IP address based on the device name. This must take place prior to system power-up. System power-up refers to the startup/restart of an automation system after ‚Power on' or after a ‚Reset'. The IP address is assigned within the same subnet using the DCP protocol integrated by default in every PROFINET IO-Device.

System power-up An IO-Controller always initiates system power-up after a startup/restart based on the configuration data without any intervention by the user. During system power-up, an IO-Controller establishes an explicitly specified communication relation (CR) and application relation (AR) with an IODevice.

Data exchange Following successful completion of system power-up, the IO-Controller and IO-Devices exchange process data, alarms, and acyclic data.

4. System Power-up

Following a ‚Power on', the following steps are performed in the field device:

- Initializing the physical interfaces in an IO-Device in order to accommodate the data traffic;
- Negotiating the transmission parameters/
- Determining the degree of expansion in the field device and communicating the information to the context management;
- Starting the exchange of the neighborhood information • Address resolution on the side of the IO-Controller;
- Establishment of communication between IO-Controller and IODevice;
- Parameterizing the submodules in the device (write records) • Retentive saving of port information to the physical device (PDev);
- Completing and checking the parameterization, and starting the data exchange

4.1 Application and communication relations

To establish communication between the higher-level controller and an IODevice, the communication paths must be established. These are set up by the IO-Controller during system startup based on the configuration data in the engineering system. This specifies the data exchange explicitly.

Every data exchange has an embedded ‚Application Relation' (AR). This establishes a precisely specified application (connection), i.e., the AR, between the higher-level controller (IO-Controller or IO-Supervisor) and the IO-Device. Within the AR, ‚Communication Relations'(CR) specify the data explicitly. An IO-Device can have multiple ARs established from various IO-Controllers.

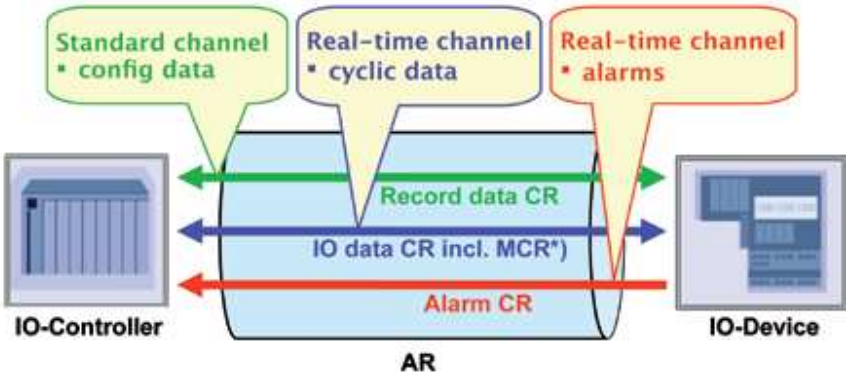

Fig. 5.1

Setting up an application relation The IO-Controller initiates setup of an application relation during system power-up. As a result, all data for the device modeling, including the general communication parameters, are downloaded to the IO-Device. At the same time, the communication channels for cyclic/acyclic data exchange (IO data CR, record data CR), alarms (alarm CR), and multicast communication relations (MCR) are set up.

Setting up a communication relation (CR) ´**C**ommunication **R**elations´ (CR) for data exchange must be set up within an AR. These specify the explicit communication channel between a consumer and a provider.

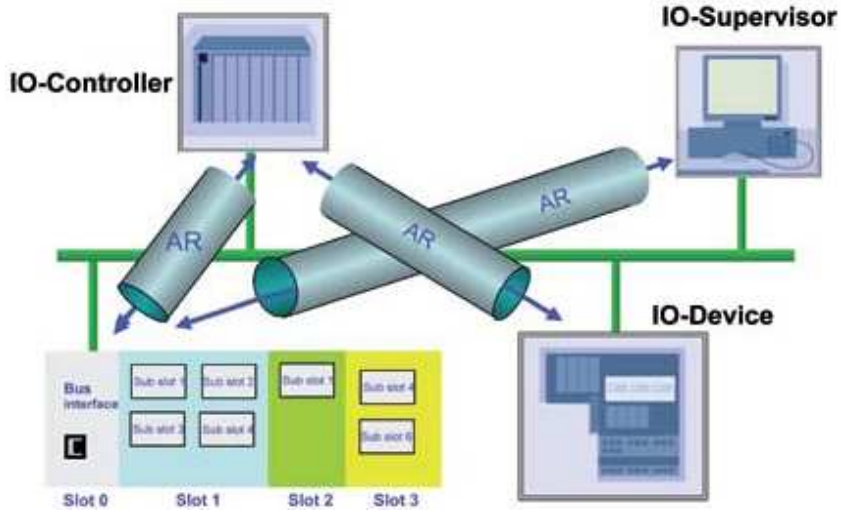

Fig. 5.2 IO Device configuration

Figure 5.2 shows an example IO Device configuration and the possible application relations with multiple controllers.

4.2 Neighborhood detection

Neighborhood detection with LLDP according to IEEE 802.1 AB and the PNO-specific additions is part of the overall concept „Device replacement without engineering tool“. This requires the ability to determine the data of neighboring devices on a port-byport basis using LLDP services and to provide these data to the higher-level controller. Together, these

conditions enable modeling of a plant topology and convenient plant diagnostics as well as device replacement without additional tools.

The **L**ink **L**ayer **D**iscovery **P**rotocol (LLDP protocol) was used to apply the principle of neighborhood detection in PROFINET IO. PROFINET field devices exchange existing addressing information with connected neighbor devices over each switch port. The neighbor devices are thereby unambiguously identified and their physical location is determined. (example in fig. 5.3: the delta device is connected to port003 of switch 1 via port001).

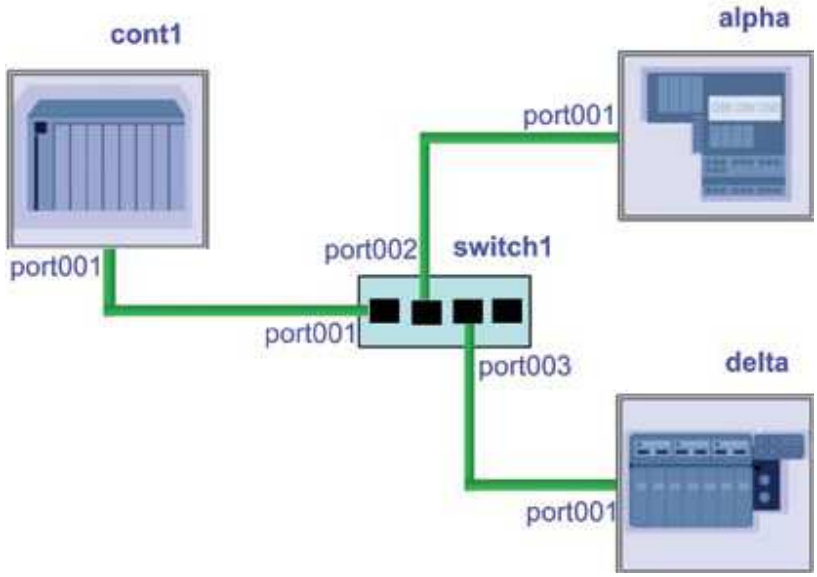

Fig. 5.3

The LLDP protocol is implemented in software and therefore requires no special hardware support. LLDP is independent of the network structure (line, star, etc.).

4.3 Topology detection

Automation systems can be configured with a line, star, or tree structure. Forthis reason, it is important to know which field devices are connected to which switch port and the identity of the respective portneighbor. The higher-level controllercan then reproduce the plant topologyaccordingly. In addition, if a field devicefails it is possible to check whether thereplacement device has been reconnected in the proper position. Plant operators also demand the ability to replace devices without an additional engineering tool. This condition can be met very conveniently through the use of PROFINET field devices.

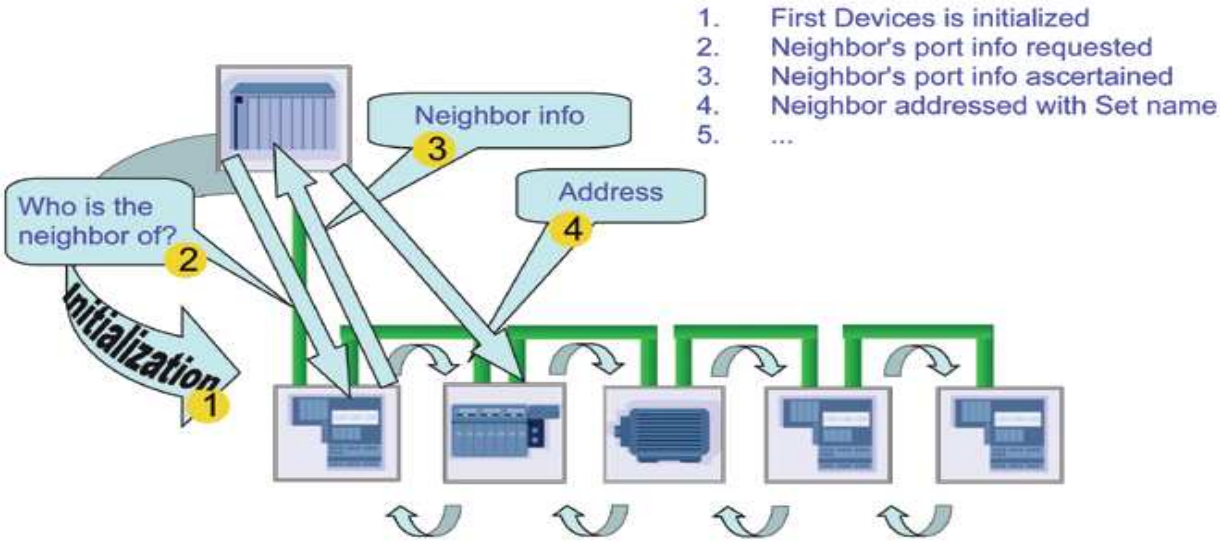

Fig. 5.4

As mentioned in the previous sections, a plant operator can use standard functionality in PROFINET to represent a plant topology and portgranular diagnostics in a graphicsbased format. This provides the plant operator a quick overview of the plant status.

4.4 Communication during connection establishment and parameterization

System power-up in an automation system with PROFINET IO is initiated by the IO-Controller. The frames presented below are always transacted via the UDP/IP channel according to the following scheme:

- **Connect frame:** Establishment of an AR and the configured CRs.
- **Write frame:** Parameterizing of all configured submodules.
- **DControl frame:** End detection of parameterization of the IO-Controller (also called ‚EndOfParameterization').
- **CControl frame:** End detection of parameterization of the IO-Device (also called ‚Application Ready').

Figure 5.5 shows the power-up sequence of an IO-Device.

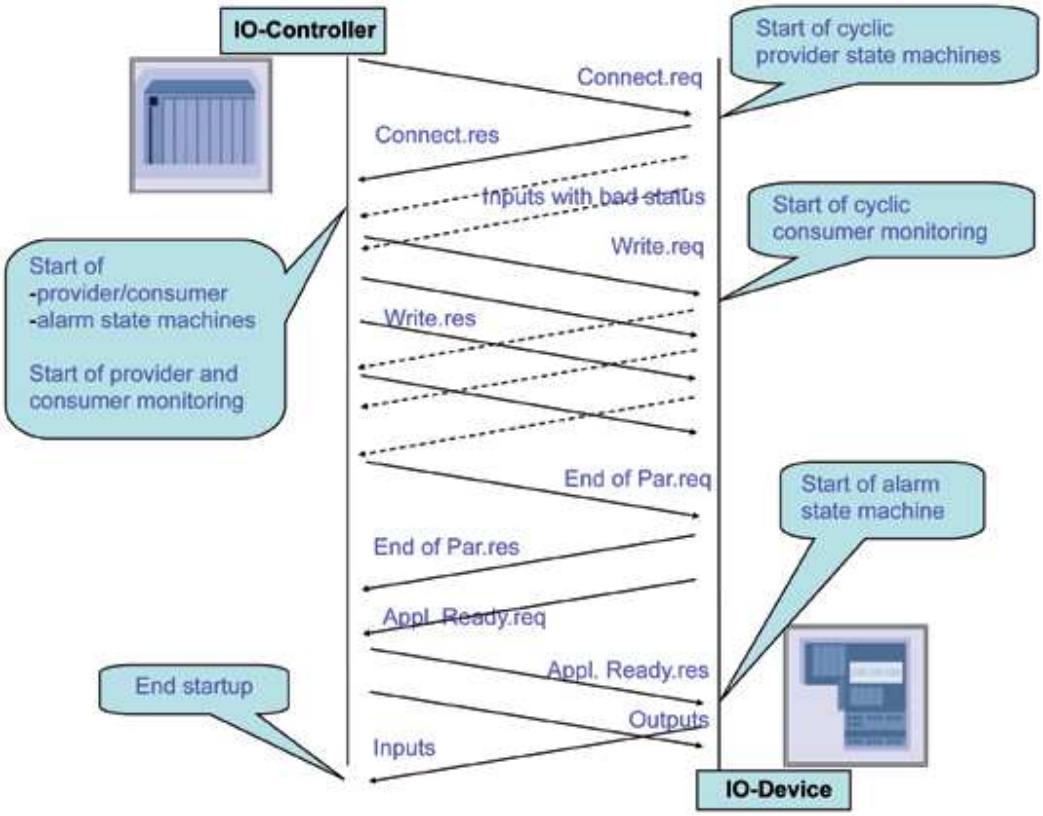

Fig. 5.5 Power-up sequence of an IO-Device

During system power-up, the following is established: cyclic I/O data, alarms, exchange of acyclic read/write services, expected modules/submodules, and any needed cross connections between IO-Devices.

The IO-Controller or IO-Supervisor uses the ‚Connect frame' to start the connection establishment and to transfer all data required to establish an AR and the necessary CRs. It contains the relevant parameterization data as well as the order, the process data traffic, and the monitoring time for the power-up. The transmission frequency of cyclic I/O data is specified during plant engineering.

The IO-Controller uses the subsequent ‚Write frames' to parameterize the configured submodules that serve as the data interface to the process.

If all parameters are loaded to the IODevice, the IO-Controller marks its end of parameterization with the ‚DControl. req' frame (‚EndOfParameterization'). The user software then creates the ultimate data structures and updates the status of the submodules.

If all data structures are created in the IO-Device and the necessary checks have been made, it sends a ‚CControl. req' to the IO-Controller to signal its readiness for productive data traffic (‚Application Ready'. From the perspective of the IO-Device, communication is now established. When the IO-Controller acknowledges the ‚Application Ready', communication is established from the perspective of the IO-Controller as well. If the IO-Device has discovered errors during parameterization, it signals these errors to the IO-Controller. The first successful exchange of I/O data marks the end of the power-up.

Following successful system powerup, the following can be exchanged:

- Cyclic process data
- Alarms
- Acyclic data traffic

In PROFINET IO, system power-up currently uses many functions that enable reliable power-up of all field devices involved in communication. This process can last several seconds until a field device is ready to operate. During a tool changeover of industrial robots, for example, these power-up times are not acceptable because the associated wait times directly affect the process (cycle time of conveyor). For this reason, the plant operator demands the ability to achieve a system power-up in significantly under 1 second. PROFINET therefore includes optimizations for fast operational readiness.

‚Fast Start Up‘ (**FSU**) is an optimized communication path for achieving data exchange in significantly less time during the second to nth power-up (for example, after re-parameterization). It is based on the fact that many parameters are already stored in the field devices. This optional path can be used in parallel to standard power-up (which is still used after power is switched on and during the first power-up or reset). This means that communication parameters must be saved retentively. The port configuration allows the user to decide whether a patch cable (1:1 wired cable) or crossed cable is to be used.

5. PROFINET IO Controller

A PROFINET IO-Controller is the station in an automation system on which the control program runs. It requests the process data (inputs from the configured IO-Devices during power-up), processes its control program, and transfers the process data to be output (outputs) to the respective IO-Devices. To perform this data exchange, it requires the system configuration data containing all the communication data. The following data are defined during system configuration:

- Degree of expansion of an IODevice;
- Parameterizations for an IODevice;
- Transmission frequency;
- Degree of expansion of the automationsystem;
- Information regarding alarms and diagnostics.

Multiple IO-Controllers can be used in a PROFINET system. If these IOControllers are to be able to access the same data in the IO-Devices, this must be specified when configuring (shared devices, shared inputs). The term „shared devices“ refers to access by multiple controllers to a single IODevice. Shared inputs describe the access by multiple controllers to the same slot in an IO-Device.

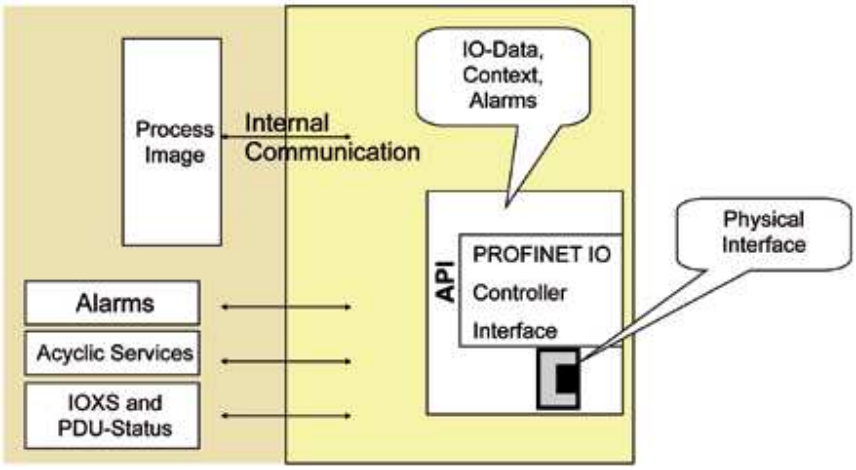

Fig. 6.1 Structural configuration of an IO-Controller

The IO-Controller receives the configuration data of the automation system and establishes the application relations and the communication relations with the configured IO-Devices autonomously. Figure 6.1 shows the structural configuration of an IO-Controller.

An IO-Controller can establish one AR each with multiple IO-Devices. Within an AR, several IOCRs and APIs can be used for data exchange. This can be useful, for example, if more than one user profile (PROFIdrive, Encoder, etc.) is involved in the communication and different subslots are required. Within an IOCR, the specified APIs (Application Process Identifier) are used for differentiation purposes. As a result, co-mingling of data between the APIs is not possible. Access to user data must be coordinated by the user program. PROFINET IO allows more than one user profile to be defined within the same AR. One IO-Controller must support the following functions:

- Alarm handling;
- Process data exchange (IO-Device in I/O area of host);
- Acyclic services;
- Parameterizing (transmitting power-up data, transmitting recipes and user parameterization of the assigned IO-Devices);
- Diagnostics of configured IODevices;
- Initiator for establishing context for an IO-Device;
- Address assignment via DCP (including the automatic detection of device failures and acceptance of a replaced field device during user data mode);
- API (Application process instance).

The parameter server functionality is available for backing up and reloading dynamic parameters of a field device. The basic parameterization of a field device is carried out using the parameters defined in the GSD file for the field device. A GSD file contains module parameters for I/O modules, among other things. These are stored as static parameters and can be loaded from the IO-Controller to an IO-Device during system power-up. For many field devices it is either impossible or inappropriate to initialize parameters using the GSD approach due to the quantities, the user guidance, or the security requirements involved. Such

data for specific devices and technologies are referred to as individual parameters (iPar). Often, they can also be specified only during commissioning. If such a field device fails and is replaced, these parameter must also be reloaded automatically to the new field device. In the past, proprietary solutions had to be used to back up and reload these parameters because no standard approach was available, i.e., the user was confronted with a wide range of solutions and operations or none at all. It was necessary to change this situation in order to offer plant operators a convenient and uniform solution.

The problem described gave rise to the so-called iPar server for saving and automatic loading of so-called dynamic device parameters. The iPar server can be viewed as an optional program section in the host/IO-Controller. It is not safety-oriented and serves to ensure that a field device stores its iParameters without additional intervention and that a replacement device can retrieve them again.

In order to better classify the function of an iPar server in the overall PROFINET context, the mode of operation will be explained using a simple application. The interaction between a TCI application, the device replacement, and the iPar server is also revealed.

During the initial commissioning, the following sequence occurs:

1. The static data from the GSD file are read into a configuring tool. General configuration is carried out as previously. In addition, the requirement for the iPar server function and the iParameter scope in Kbytes can now be declared from special GSD parameters.
2. During system power-up (connect and write services), an IO-Controller initializes the field device with the data generated from the GSD in order to place the field device in data exchange and prepare it for acyclic communication.
3. By means of a parameterization tool, the iParameters can now be assigned for the corresponding field device via a user dialog. It‘s up to the device manufacturer to determine the approach by which the iParameters reach the device, e.g., via an interface such as TCI (Tool Calling Interface), direct point-to-point connection such as RS 232, infrared connection, or local teach-in.
4. The most convenient approach involves connection of the parameterization tool to the field device via an interface such as TCI using the TCI Communication Server.
5. Following the iParameter assignment and, if necessary, verification, the field device answers with an alarm notification and requests an upload of the iParameters.
6. The iPar server in the host/IOController reads the iParameters from the field device acyclically and saves them so that they can be reloaded if required (e.g., during device replacement).

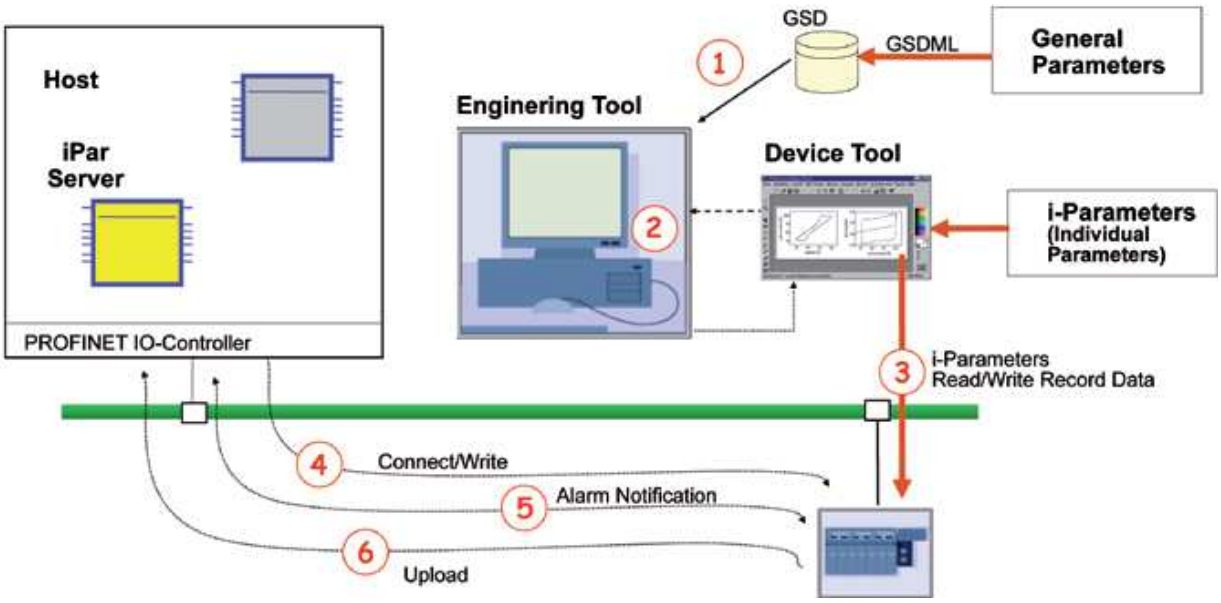

Fig. 6.2

Following device replacement and subsequent power-up (connect and write sequences), the IO-Controller loads all GSD-based data in order to place the replaced field device into data exchange again. After the basic initialization, the replaced field device determines that it still needs iParameters. An alarm notification is then generated (Update & Retrieval alarm). The iPar server then loads the saved iParameters of the predecessor device to the field device.

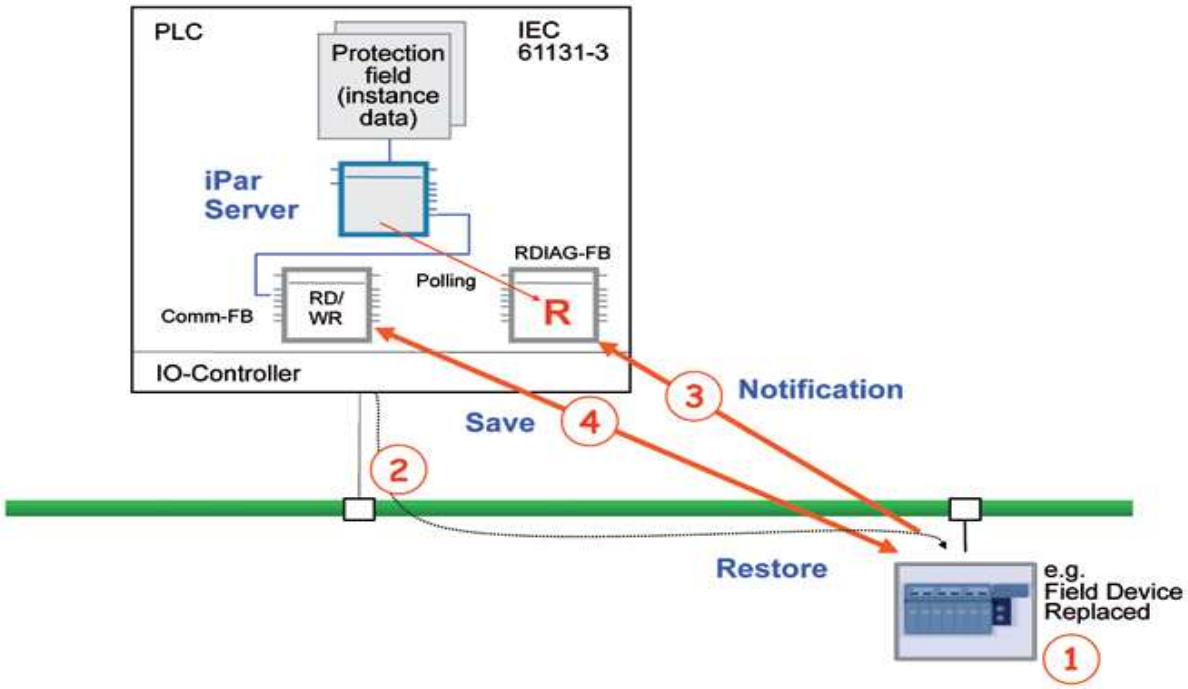

Fig. 6.3

The detailed chronological sequence is as follows:

1. The defective field device is replaced, and the replacement device is switched on.
2. Following the automatic address resolution, the IO-Controller establishes an AR with the field device and loads the GSD-based static parameters.
3. The new field device determines that it still needs (dynamic) i-Parameters and signals this via an alarm notification (´iPar request´).
4. The iPar server loads the requested iParameters into the new field device.

7. Device Description (GSD file)

The functionality of a PROFINET IODevice is always described in a GSD file. This file contains all data that are relevant for engineering as well as for data exchange with the IO-Device.

PROFINET IO-Devices can be described using XML-based GSD. The description language of the GSD file, i.e., GSDML (**G**eneral **S**tation **D**escription **M**arkup **L**anguage) is based on international standards. As the name suggests, the GSD file is a language-independent XML file (e**X**tensible **M**arkup **L**anguage). Many XML parsers are currently available on the market for interpreting XML files.

Every manufacturer of an PROFINET IO-Device must supply an associated GSD file according to the GSDML specification. This file is tested as part of certification testing.

To describe PROFINET IO-Devices, PI provides an XML schema to each manufacturer. This allows a GSD file to be created and tested easily. The need for numerous subsequent input checks is therefore omitted.

In addition, the device model of PROFINET IO exhibits a further hierarchy level for data addressing when compared to PROFIBUS. Thus, for example, addressing within a field device (in PROFIBUS: slot and index) has been expanded to include the identifier of a subslot. In PROFINET IO, addressing within a field device can be performed with finer granularity (slot and subslot). In addition, this type of addressing could not be described with the GSD file for PROFIBUS.