PROFIBUS Integration in PROFINET IO

Amendment 1

to

Fieldbus Integration into PROFINET IO

Version 1.0

Guideline

Version 1.0
September 2006

Order No: 7.012

Document identification: TC2-06-0002a

File name: PROFIBUS-Integration_7012_V10_Sep06

Prepared by the PROFIBUS Working Group 9 “Fieldbus Integration” in the Technical Committee 2 “Communication Profiles”.

The attention of adopters is directed to the possibility that compliance with or adoption of PI (PROFIBUS International) specifications may require use of an invention covered by patent rights. PI shall not be responsible for identifying patents for which a license may be required by any PI specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. PI specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

NOTICE:

The information contained in this document is subject to change without notice. The material in this document details a PI specification in accordance with the license and notices set forth on this page. This document does not represent a commitment to implement any portion of this specification in any company's products.

WHILE THE INFORMATION IN THIS PUBLICATION IS BELIEVED TO BE ACCURATE, PI MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR PARTICULAR PURPOSE OR USE.

In no event shall PI be liable for errors contained herein or for indirect, incidental, special, consequential, reliance or cover damages, including loss of profits, revenue, data or use, incurred by any user or any third party. Compliance with this specification does not absolve manufacturers of PROFIBUS or PROFINET equipment, from the requirements of safety and regulatory agencies (TÜV, BIA, UL, CSA, FCC, IEC, etc.).

PROFIBUS® and PROFINET® logos are registered trademarks. Their use is restricted for members of Profibus International. More detailed terms for their use can be found on the PROFIBUS website at www.profibus.com/libraries.html under "Presentations & logos".

In this specification the following key words (in **bold** text) will be used:

may: indicates flexibility of choice with no implied preference.

should: indicates flexibility of choice with a strongly preferred implementation.

shall: indicates a mandatory requirement. Designers **shall** implement these mandatory requirements to ensure interoperability and to claim conformance with this specification.

Publisher:
PROFIBUS Nutzerorganisation e.V.
Haid-und-Neu-Str. 7
76131 Karlsruhe
Germany
Phone: +49 - 7219 - 65 85 90
Fax: +49 - 7219 - 65 85 89
E-mail: info@profibus.com
Website: www.profibus.com

Contents

List of Figures

List of Tables

Revision Log

Identification	Version	Originator	Date	Change Note/History/Reason
RGJ	0.1	TC2 WG 9	2006/03/06	First draft
ZH	0.2	TC2 WG 9	2006/04/06	Supplementations and corrections
RGJ	0.3	TC2 WG 9	2006/04/17	Adjustments after internal review
RGJ	0.4	TC2 WG 9	2006/05/17	Supplementations after reject by PI board
RGJ	0.5	TC2 WG 9	2006/05/24	Revision after WG meeting
ZH	0.6	TC2 WG 9	2006/04/31	Supplementations and corrections
DZ	0.7	TC2 WG 9	2006/06/09	Revision after WG meeting
ZH, RGJ	0.8	TC2 WG 9	2006/06/18	Further revision according to comments of WG members
RGJ	0.9	TC2 WG 9	2006/06/19	Joker block additions, module & submodule fixings
DZ, RGJ	0.91	TC2 WG 9	2006/06/21	Refinement of Prm command composing, inclusion of miscellaneous review comments
RGJ	0.92	TC2 WG 9	2006/06/22	Error mending
RGJ	0.93	TC2 WG 9	2006/06/26	Minor adjustment according internal discussion
RGJ	0.94	TC2 WG 9	2006/07/06	Revision after WG internal review
DZ	0.95	TC2 WG 9	2006/08/10	Revision after WG meeting
TC2-06-0002a	1.0	TC2WG9	2006/09/22	Final document by decision of advisory board

1 Management Summary – Scope of this Document

PROFIBUS is mainly used at the field level with interfacing capabilities downward to the sensor/actor level as well as upwards to the production and enterprise levels. The PROFIBUS implementation is applicable and well established in the segment of the manufacturing and process industries as well. PROFINET is the solution of PROFIBUS International for the use of Ethernet in fieldbus technology. It provides a common solution for industrial communication. Because of this, a basic step is the integration of PROFIBUS together with other existing fieldbusses in PROFINET as a widely accepted common industrial Ethernet standard.

The mapping of the PROFIBUS wire is defined onto the logical sight of PROFINET IO that will offer the following benefits:

- Customers can use their existing PROFIBUS expertise and experience when they decide to use PROFINET IO.
- Some profiles and functionalities, especially for process automation, are not yet provided by PROFINET IO.
- PROFIBUS devices offer widely spread solutions in the manufacturing and process industry. The numeration of PROFINET IO devices is continuously increasing but at the moment not all applications can be covered by PROFINIET IO devices. So the combination of PROFIBUS and PROFINET IO will offer an ideal solution in many customer applications.

The mapping described in this guideline has to be reconceived installing PROFIBUS in process automation and expanded accordantly. Another increment may be a determination of additional diagnosis mapping.

This guideline describes the basic concepts for the integration of PROFIBUS systems into PROFINET IO and is based on the main document “Fieldbus Integration into PROFINET IO”. All definitions below are valid both for PROFIBUS DP and PROFIBUS PA the following aspects are considered:

- Cyclic process data, acyclic parameterisations
- Diagnostics
- Alarms
- Start-up behaviour
- Engineering

Not considered are:

- An establishment of an end-to-end relationship of IRT applications.
- Sync / Freeze handling

The implementation of any PROFIBUS feature not specified in this document is left to the manufacturer.

The following diagram provides a system overview for PROFIBUS and PROFINET IO.

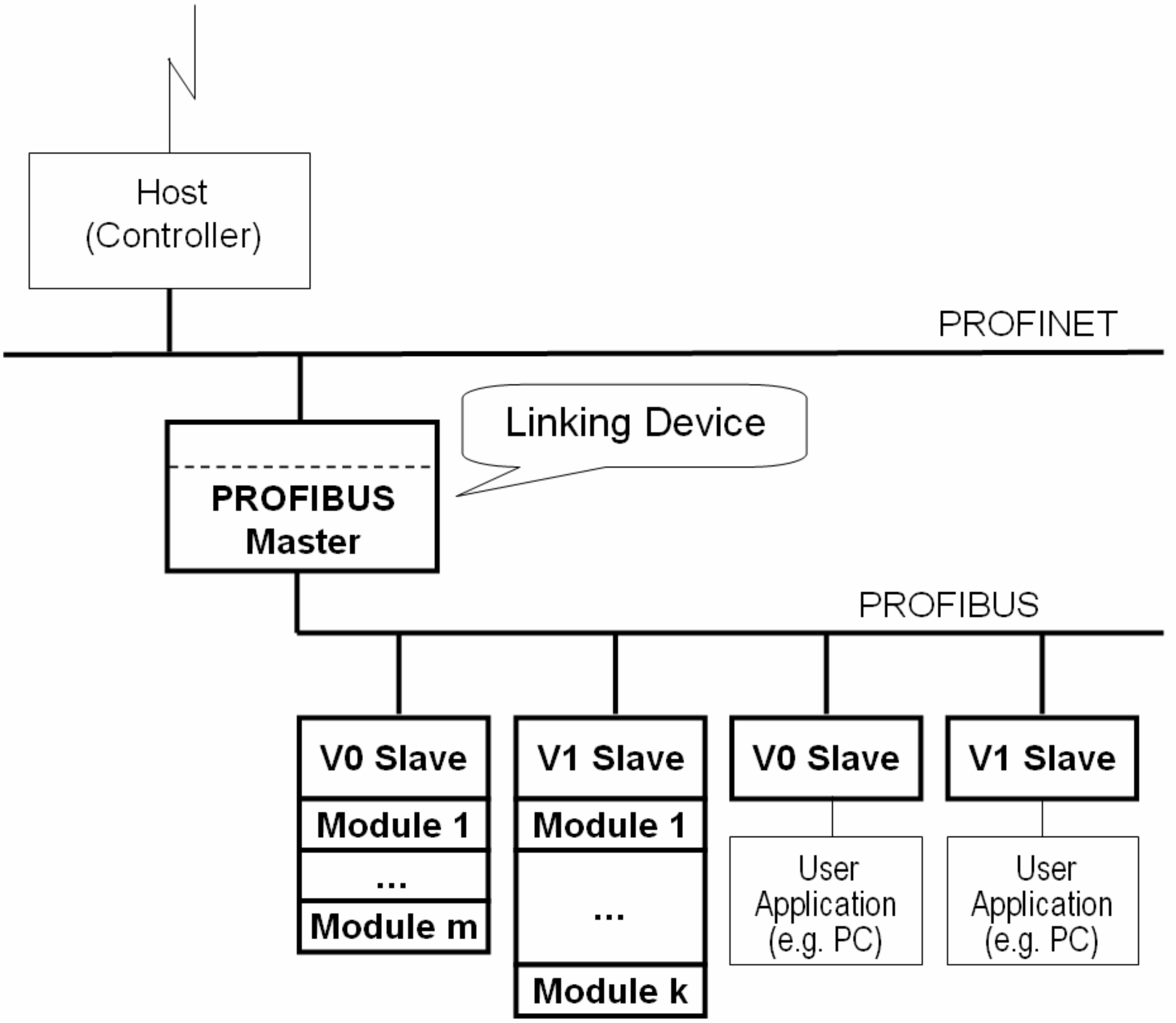


Figure 1: System overview

The definitions for PROFIBUS integration are standardized. It is therefore possible to create interoperable and replaceable Linking Devices that link PROFIBUS to PROFINET IO.

2 List of affected patents

There is no affected patent known by the members of TC2WG9, Fieldbus Integration. The list is empty. No patent search, neither external nor internal, has been done by the members of the Working Group up to now. PROFIBUS International does not guarantee the completeness of this list.

3 Related Documents and References

References

- PROFIBUS glossary (www.profibus.com)

[1] IEC 61158/ 61784, Type 10 (draft): Digital data communication for measurement and control – Fieldbus for use in industrial control systems (Type 10: PROFINET)

[2] IEC 61158/ 61784, Type 3: Digital data communication for measurement and control – Fieldbus for use in industrial control systems (Type 3: PROFIBUS)

[3] Fieldbus Integration into PROFINET IO, Draft 0.5

4 Definitions and Abbreviations

4.1 Definitions

Linking Device	A PROFINET IO device with integrated PROFIBUS master functions. It is used to link PROFIBUS to PROFINET IO
Mapping Application	Software running in Linking Device implementing the coupling between PROFINET IO and PROFIBUS
PROFIBUS Master System	Profibus Master Segment driven by Linking Device
DeviceModule	PROFINET Module representing the whole device ("DAP")
PBMasterModule	Module representing a PROFIBUS master
PBSlaveModule	Module representing a PROFIBUS slave
DeviceSubmodule	Submodule containing Device data
PBMasterSubmodule	Submodule in PBMasterModule
PBSlaveSubmodule	Submodule in PBSlaveModule representing whole PROFIBUS slave
PBModuleProxySubmodule	Submodule in PBSlaveModule representing a PROFIBUS module

4.2 Abbreviations

AR	Application Relationship *Relationship between PROFINET-IO-Controller and PROFINET-IO-Device or between PROFINET-IO-Supervisor and PROFINET-IO-Device* *IO-AR: Relationship between PROFINET-IO-Controller and PROFI-NET-IO-Device* *Supervisor AR Relationship between PROFINET-IO-Supervisor and PROFINET-IO-Device*
APDU	Application Protocol Data Unit *By the APDU, cyclic data are transferred. It contains the flags "Data Flag", "AR State Flag", "Provider State Flag", and "Problem Indicator Flag".*
API	Application Process Identifier *Every submodule contains one ore several special application processes referenced by their identifier (API).*
Data Record	Data unity *Is read or written acyclic by the IO-Controller. The data record lies on a submodule and is addressed by an index.*
GSDML	Generic Station Description Markup Language, *Used in this document as PROFINET Device Description File*
GSD	*Used in this document as PROFIBUS Device Description File*

5 Responsibilities

Not defined.

6 Introduction

6.1 PROFIBUS Configuration in PROFINET

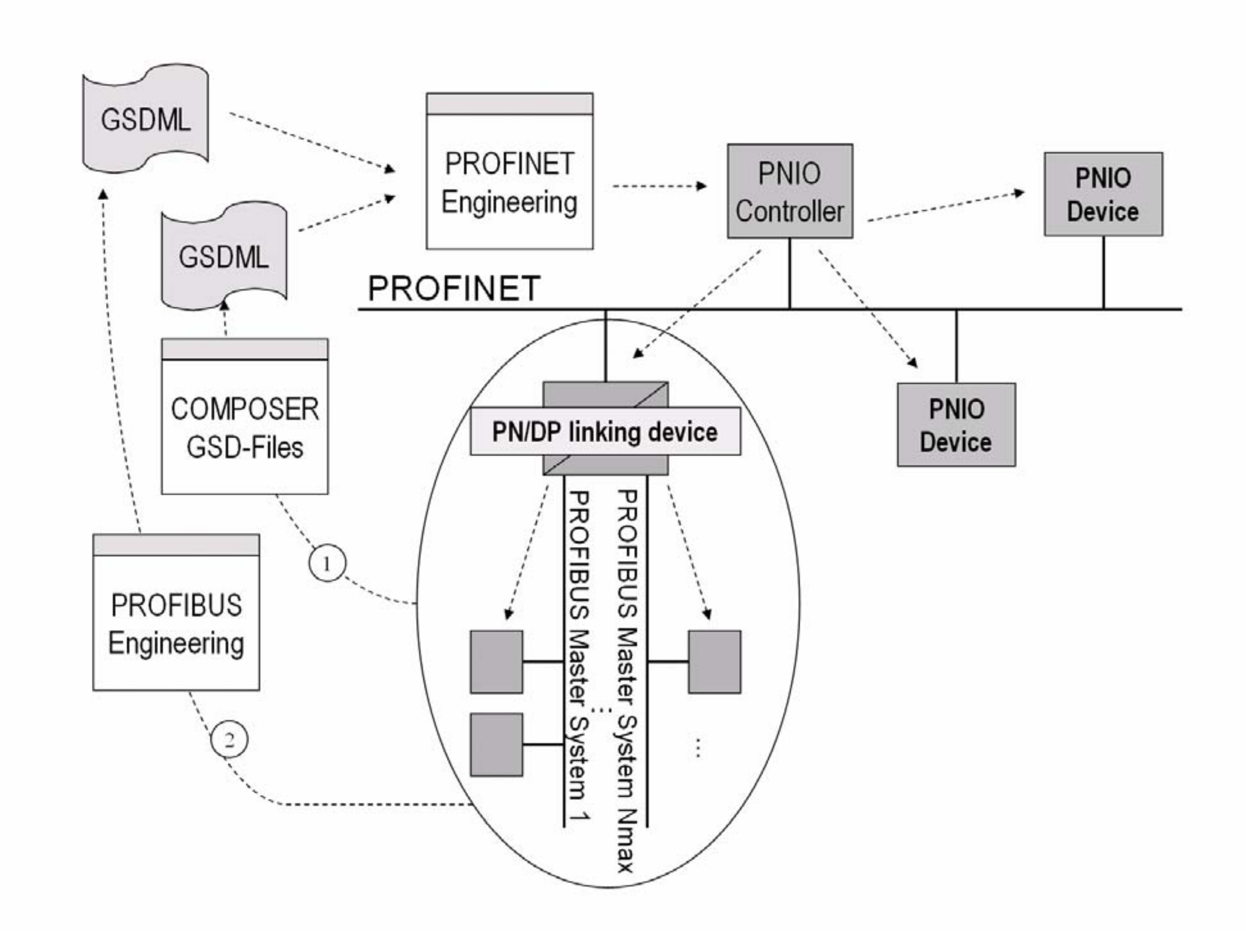


Figure 2: Integration of PROFIBUS in PROFINET IO

For coupling PROFIBUS slaves to PROFINET-IO the controller's engineering uses a dynamically generated GSDML .There are two ways for this GSDML formation:

1. The PROFINET GSDML is build from GSD files (typical way).
2. GSDML is built from the contents of a PROFIBUS engineering project.

The exported GSDML will be the input of a PROFINET engineering including all descriptions of PROFIBUS devices needed in the PROFINET engineering context. After dropping a PN/PROFIBUS Linking Device on the graphical surface of the PROFINET-IO controller's engineering tool into an engineering project it depends of the generating tool whether additional engineering is necessary.

If the bus had been configured totally in the PROFIBUS engineering tool, only the download to the PROFINET IO controller is to be initiated after the user completed planning of the IO controller and other PNIO devices.

Nevertheless, if not fixed, the PROFIBUS slaves may be parameterised and configured in the PROFINET-engineering according to their requests. For this purpose the slots of the Linking Device have to be filled with suitable modules. For that reason several modules are offered by GSDML. Each module represents one PROFIBUS slave, the engineering displays definite PNIO start-up data records for parameterisation and configuration that may be changed by the user. Finally, the user-completed planning is loaded together with that of other PNIO devices onto the controller.

The PROFINET-IO controller loads the configuration and the parameter data into the IO devices by sending the PROFINET-IO data records to the IO devices.

Subsequently, if any configuration or parameter data have been loaded, the PROFIBUS Master(s) situated in the Linking Device parameterises its depending PROFIBUS slaves according to the received parameterisation record data.

6.2 Structure of the Linking Device

The general structure of a PN/PROFIBUS Linking Device based on the described mapping model below is shown in Figure 3. The defined mapping structure is to be implemented in the “Mapping Application” block. The access to Ethernet or to the 1 up to N_{max} PROFIBUS Master System may be carried out by a standard software packet or by an ASIC with appropriate functionality.

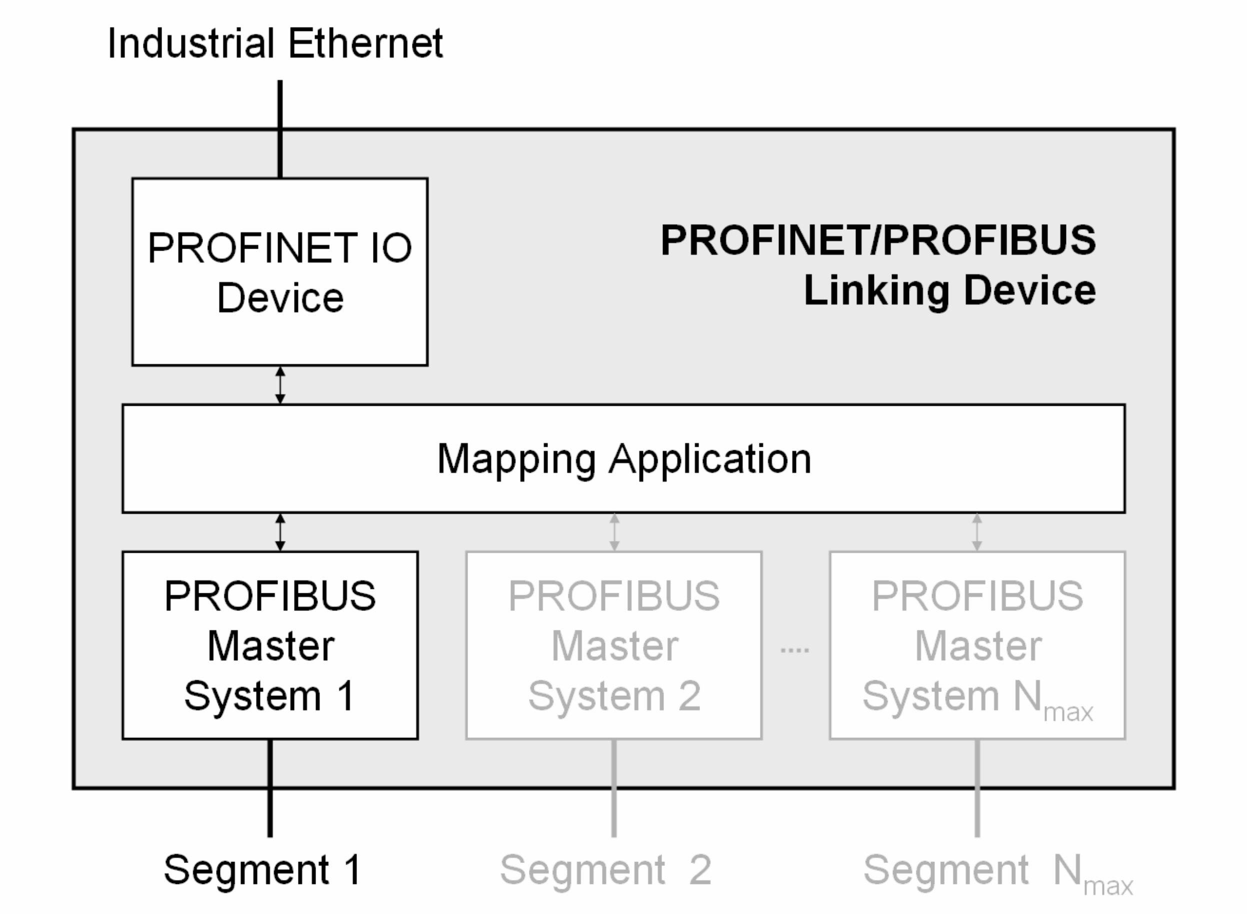


Figure 3: Internal structure of the PN/PROFIBUS Linking Device

7 Mapping Model

7.1 Modular Mapping Model

The integration of PROFIBUS in PROFINET IO is done by means of the modular mapping. In this model, every PROFIBUS slave has a corresponding module in a PROFINET IO device according to the slot/subslot addressing model [1]. The device description of the PROFINET IO device contains all possible PROFIBUS slaves with their device parameters.

Modular mapping offers the following advantages:

+ **Transparent engineering:**
  Every PROFIBUS slave is visible as a module in the engineering tool of the PROFINET IO controller. The device description is a generic module description in the GSDML.

+ **Good data availability information:**
  The data of all PROFIBUS modules can be transferred in a single PROFINET IO frame. Additionally any subslot adds an IOPS/IOCS that show the actual data availability of at least slave granularity.

+ **Modular slaves:**
  The modules of a PROFIBUS slave can be mapped to PROFINET IO submodules. All handling necessities are offered by today’s engineering systems.

The specification team decided to go for specification of the modular mapping. This mapping method is therefore described further within this document.

7.2 Supervisor Usage

Because of the mapping of real PROFIBUS slaves and PROFIBUS Master System to PROFINET IO modules and submodule there is a dependency between some of these particular elements. So an IO Supervisor may not takeover all kinds of modules as stated below. These restrictions will be described in chapter 8.9 “Dynamic Behaviour” and result in directions of stetting certain parameters in the ModuleDiff-Block.

Modules without permission for takeover are explicitly noted in 8.9.2 and **shall** be handled accordingly in the SubmoduleState (see Table 14 and Table 17): The ‘AddInfo’ field **shall** be set to ‘Takeover is not allowed’.

For acyclic access (PROFIBUS MSAC2 Connections) from a parameterisation tool, no additional support by the Linking Device is necessary. The supervisor can use a Supervisor AR with DeviceAccess where no additional knowledge about the modules and submodules (especially their identification numbers) is required. The PROFIBUS Slot/Index record data is mapped transparently to PROFINET subslot/index record data. Only READ and WRITE services (no PROFIBUS DataTransport) **should** be supported.

The strategy on when to open and close a C2-connection to a slave is manufacturer specific.

State table 324 “State table for a submodule” [1] states that Read/Write requests to a non-existing module shall be answered with a negative response. In actual Linking Device implementations, it may happen that Read/Write accesses to PBModuleProxySubmodules are possible that have no corresponding module and submodule entry in the context management. This happens because IOARs with “DeviceAccess” bypass the context management in the stack. The resulting contradiction is that ReadRealConfiguration-Data reports a non-existing submodule although Read/Write accesses to this submodule are possible. This is not considered an issue.

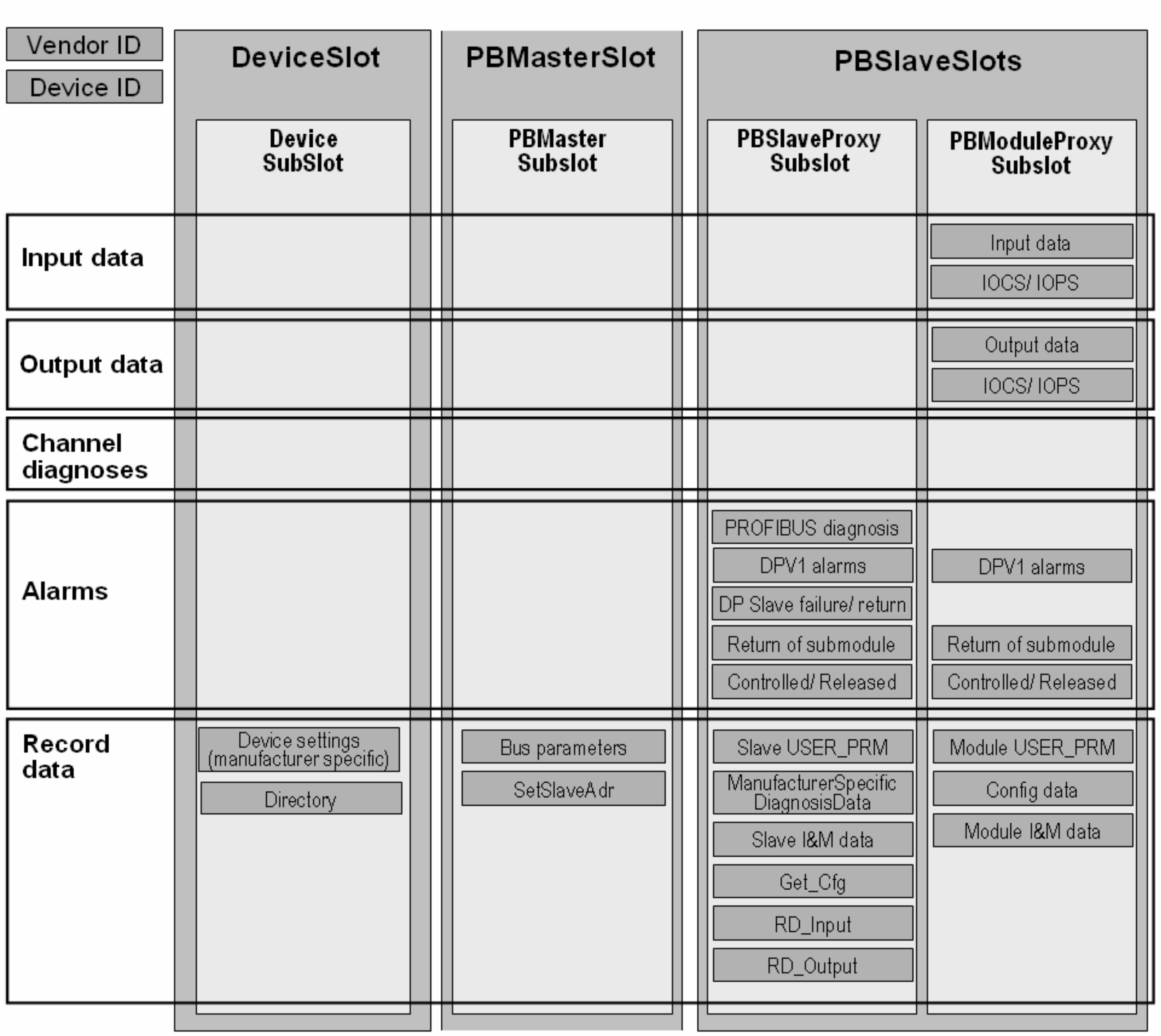
Because definition of the IO Supervisor performance is in progress at the moment, no further constraints are defined concerning the PROFIBUS mapping to PROFINET IO.

8 Slot/Subslot Mapping

8.1 Overview

8.1.1 Device Layout

The following diagram illustrates PROFIBUS mapping to the slot/subslot-addressing model of PROFINET IO. This diagram forms the basis of this document. All subsequent sections refer to parts of this diagram.



	DeviceSlot	PBMasterSlot	PBSlaveSlots	
	Device SubSlot	PBMaster Subslot	PBSlaveProxy Subslot	PBModuleProxy Subslot
Input data				Input data IOCS/ IOPS
Output data				Output data IOCS/ IOPS
Channel diagnoses				
Alarms			PROFIBUS diagnosis DPV1 alarms DP Slave failure/ return Return of submodule Controlled/ Released	DPV1 alarms Return of submodule Controlled/ Released
Record data	Device settings (manufacturer specific) Directory	Bus parameters SetSlaveAdr	Slave USER_PRM ManufacturerSpecific DiagnosisData Slave I&M data Get_Cfg RD_Input RD_Output	Module USER_PRM Config data Module I&M data

Figure 4: Mapping of PROFIBUS properties to the slot/subslot model of PROFINET IO

8.1.2 PROFIBUS/PROFINET Mapping Overview

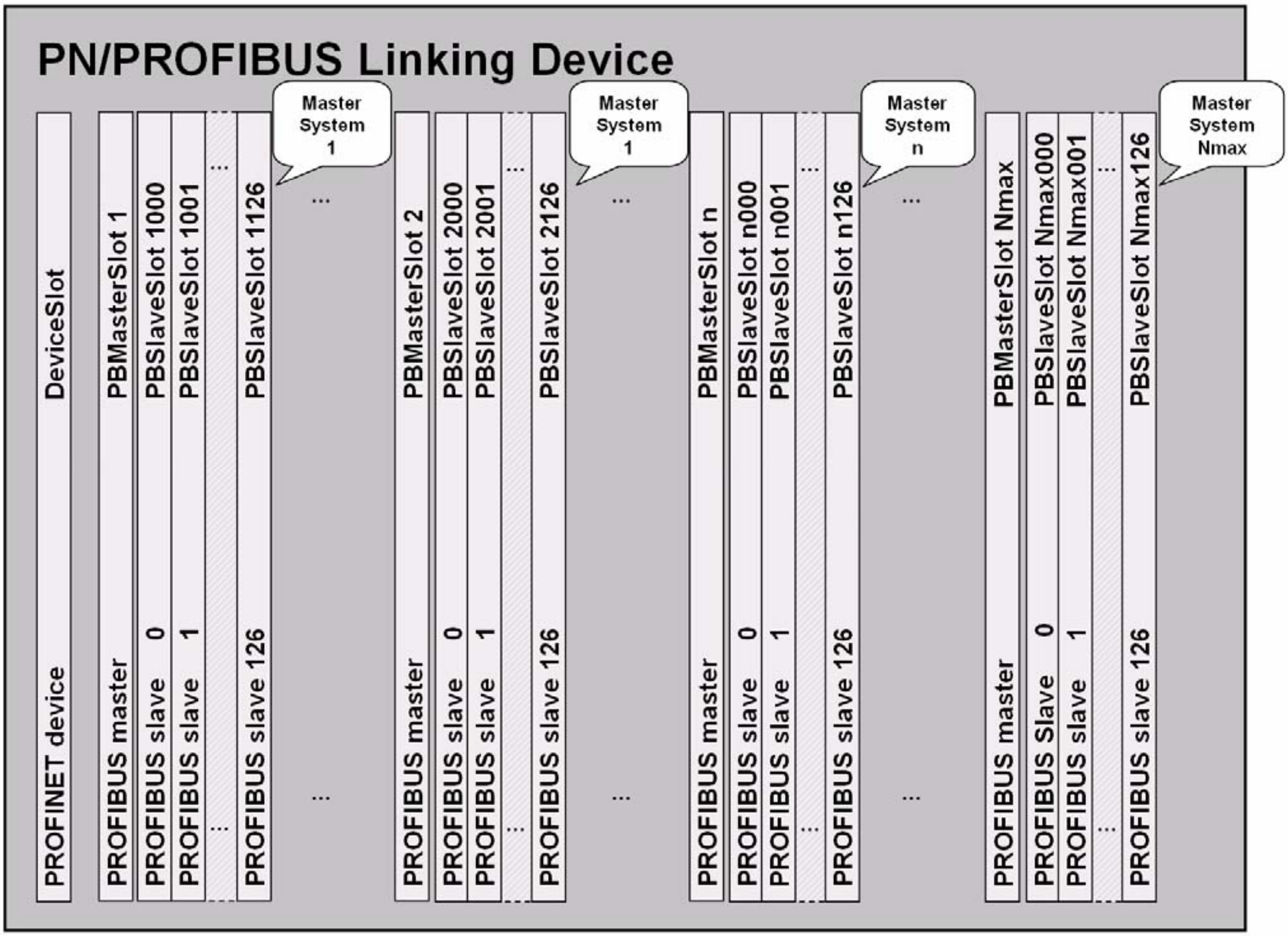


Figure 5: Modelling of a PROFIBUS Configuration for a controller application

Being a comparatively easy composed connection possibility to PROFINET-IO a PROFIBUS Master System will be mapped by the modular model:

The PROFIBUS slaves **shall** be represented as modules of a PROFINET-IO device. The *PBSlaveModule* module **shall** be placed in a *PBSlaveSlot*.

The *DeviceModule* placed in the *DeviceSlot* **shall** represent the Linking Device. It is additionally the device access point (DAP) of the PROFINET IO device.

The PROFIBUS Master Systems **shall** have their own proxy modules (*PBMasterModule*) in the *PBMasterSlots*. These objects **may** report manufacturer specific alarms for each PROFIBUS Master System that refer to the current master or the total device as a manufacturer specific implementation.

8.1.3 Fieldbus Elements Overview

The relevant fieldbus elements of PROFIBUS are given in the table below:

	Slot	**Subslot**	**Identification**	**Output data and status**	**Input data and status**	**Alarms**	Records	Channel diagnosis
Ethernet interface	*DeviceSlot*	0x8000, 0x8001, ...	PROFINET V2					Port errors/ statistics
Bus parameters for PROFIBUS Master System	*PBMasterSlot*	PBMasterSubslot	№ of PROFIBUS Master System				Part of GSD	
Slave Input/Output data	*PBSlaveSlot*	PBModuleProxySubslot	PROFIBUS Master System № & PROFIBUS address	Usage	Usage			
PROFIBUS slave diagnoses	*PBSlaveSlot*	PBSlaveProxySubslot	PROFIBUS Master System№ & PROFIBUS address			Usage		
PROFIBUS Prm telegram	*PBSlaveSlot*	PBSlaveProxySubslot and PBModuleProxySubslot	PROFIBUS Master System № & PROFIBUS address				Part of GSD	
PROFIBUS Config telegram	*PBSlaveSlot*	PBModuleProxySubslot	PROFIBUS Master System № & PROFIBUS address				Part of GSD	
C2 Channel	*PBSlaveSlot*	PBSlaveProxySubslot and PBModuleProxySubslot	PROFIBUS Master System № & PROFIBUS address				Usage	

Table 1: Functional mapping of PROFIBUS elements to PROFINET

8.2 Linking Device Structure

The definitions in the following subchapters **shall** be implemented to all PN/ PROFINET linking device.

8.2.1 Slot Layout For Device

Slot Number	Symbolic Name	Description
0	DeviceSlot / DeviceModule	Device
1 … N_{max}	PBMasterSlot / PBMasterModule	PROFIBUS Master System 1 to N_{max}, $N_{max} \leq 32$
1000 … 1126	PBSlaveSlot / PBSlaveModule	PROFIBUS Master System 1 with Slave x, $0 \leq x \leq 126$.
…		
n *1000 … n *1000 +126	PBSlaveSlot / PBSlaveModule	PROFIBUS Master System n with Slave x, Slot Number = n *1000 +x, $0 \leq x \leq 126$, $1\leq n \leq N_{max}$
Others	Manufacturer specific use	

Table 2: Slot layout for PN/ PROFINET linking device

8.2.2 Subslot Layout For DeviceModule

Subslot	Symbolic Name	Index	Symbolic Name
0x0000	**Shall** not be used		
0x0001	DeviceSubslot / Device-Submodule	0x0000	PBDirectoryRecordData
		0x0001-0x0FFF	Reserved for profile specific use
		0x1000-0xFFFF	Reserved for manufacturer specific use
0x0002-0x0FFF	Reserved for profile specific use		
0x1000-0x7FFF	Reserved for manufacturer specific use		

Table 3: Subslot layout for DeviceModule

8.2.3 Subslot Layout For PBMasterModule

Subslot	Symbolic Name	Index	Symbolic Name
0x0000	**Shall** not be used		
0x0001	PBMasterSubslot / PBMasterSubmodule	0x0000	PBMasterParametersRecordData
		0x0001	PBSetSlaveAddressRecordData
		0x0002-0x0FFF	Reserved for profile specific use
		0x1000-0xFFFF	Reserved for manufacturer specific use
0x0002	PBSlaveSettingsSubslot / PBSlaveSettingsSubmodule	0x0000-0x007E	PBSlaveInitiateRecordData
		0x007F-0x0FFF	Reserved for profile specific use
		0x1000-0xFFFF	Reserved for manufacturer specific use
0x0003-0x0FFF	Reserved for profile specific use		
0x1000-0x7FFF	Reserved for manufacturer specific use		

Table 4: Subslot layout for PBMasterModule

8.2.4 *Subslot Layout For PBSlaveModule*

Subslot	Symbolic Name	Index	Symbolic Name
0 .. 999	**Shall** not be used		
1000	PBSlaveProxySubslot / PBSlaveProxySubmodule (corresponds to PROFIBUS slave slot 0)	0x0000 - 0x00FF	PBSlaveRecordData
		0x0100	PBSlavePRMRecordData
		0x0101	PBGetConfigRecordData
		0x0102	PBReadInputRecordData
		0x0103	PBReadOutputRecordData
		0x0104-0x0FFF	Reserved for profile specific use
		0x1000-0xFFFF	Reserved for manufacturer specific use
1001 . 1254	PBModuleProxySubslot / PBModuleProxySubmodule (corresponds to PROFIBUS slave slot 1…254)	0x0000 - 0x00FF	PBSlaveRecordData
		0x0100	PBModulePRMRecordData
		0x0101	PBConfigRecordData
		0x0105-0x0FFF	Reserved for profile specific use
		0x1000-0xFFFF	Reserved for manufacturer specific use
0x04E7-0x0FFF	Reserved for profile specific use		
0x1000-0x7FFF	Reserved for manufacturer specific use		

Table 5: Subslot layout for PBSlaveModule

8.3 Structure of The Module Identification Number

The ModuleIdentNumber **shall** have the following structure:

Bit 31..24	Bit 23..16	Bit 15..0	Description
Type = 0x01	0x000000		DeviceModule for PROFINET V1.1 controllers
Type = 0x02	0x000000		DeviceModule for PROFINET V2.0 controllers
Type = 0x03	0x000000		PBMasterModule
Type = 0x04	0x00	PROFIBUS Ident№	PBSlaveModule
Type = 0x05-0x7F	*<reserved>*		Reserved for profile specific use
Type = 0x80-0xFF	*<manufacturer specific>*		Reserved for manufacturer specific use

Figure 6: Structure of the ModuleIdentNumber

8.4 Structure of The Submodule Identification Number

The SubmoduleIdentNumber **shall** have the following structure:

Module	Bit 31..16	Bit 15..0
Vendor specific use at each module	0x0000-0xFFFE	<manufacturer specific>
PBSlaveModule	0xFFFF	Unique number. If available PROFIBUS module identification from GSD **should** be used.
PBMasterModule	0xFFFF	0x0000: Master module as described in this specification 0x0001-0xFFFF: reserved
DeviceModule	0xFFFF	0x0000: DAP for PROFINET V1.1 controllers 0x0001: DAP for PROFINET V2.x controllers 0x0002-0xFFFF: reserved

Figure 7: Structure of a SubmoduleIdentNumber

8.5 Device Module

The DeviceSubmodule in the DeviceModule contains a directory to read the actual layout of the PBMasterModules and PBSlaveModules. A supervisor should read the directory in order to find out the actual device layout before accessing a PROFIBUS slave through the Linking Device.

In the following the ordering of transmission of data type values within all record data exceeding one byte length is carried out according to the conventions in [1]. That means if the RPC Flag DRep (Little Endian or Big Endian) is part of the RPCHeader or NDREPMapPDU. If it is part of the NDRDataxxx PDUs then only Big Endian Format is used.

8.5.1 PBDirectoryRecordData

Byte	**Description**
0	PROFIBUS type: 0x00: PROFIBUS DP 0x01: PROFIBUS PA 0x02...0x7F: reserved for profile specific use 0x80...0xFF: manufacturer specific
1	Number of PROFIBUS Master Systems (N_{max} “PROFIBUS Lines”)
2	PBSlaveSlot Offset
3	
4	PBSlaveSlot Multiplicator
5	
6	Beginning of SlaveProxy Subslots (PBSlaveProxySubslot)
7	
8 ... N_{max} +7	List of Slot Numbers of PBMasterModules

Figure 8: Structure of “PBDirectoryRecordData”

The position of a *PBSlaveModule* can be calculated as follows:

PROFINET SlotNumber = PBSlaveSlot Offset +(PBSlaveSlot Multiplicator *(n -1)) +PROFIBUS SlaveAddress, where n is the number of the PROFIBUS Master System.

The position of the *PBModuleProxySubslot* can be calculated as follows:
PROFINET SubslotNumber = Position of PBSlaveProxySubslot + PROFIBUS Slot Number (0 .. 254).

The position of the *PBMasterModules* can be read directly in the structure.

8.6 PROFIBUS Master Module

Additionally to the specification below, the *PBMasterSubmodules* in the *PBMasterModule* **may** be capable of

- Receiving manufacturer specific parameterisation by record data
- Manufacturer specific implemented channel diagnoses signalled by the PROFIBUS masters

8.6.1 Modelling of the PROFIBUS Master Systems

PROFIBUS masters are modelled as PROFINET modules (*PBMasterModule*). Each *PBMasterModule* contains a PBMasterSubmodule that carries the master's parameterisation record data.There is no correlation between slot or subslot number and the bus address the PROFIBUS Master uses.

The *PBMasterModule* and *PBMasterSubmodule* identification numbers are specified in 8.3 and 8.4.

The *PBMasterSubmodule* of the *PBMasterModule* **shall** support elements of following PROFINET IO ASEs:

- Data records details see 8.6.2.

Elements not specified in this document are:

- Alarms
- Channel Diagnosis

The manufacturer **may** implement these services as an extension.

8.6.2 PROFIBUS Bus Parameter (PBMasterParameterRecordData) Record Data

The PROFIBUS bus parameter record data **shall** be fixed according Figure 9.

Byte №	Description
0	Major Version = 0x00
1	Minor Version = 0x00
2	TS
3	Data Rate
4	T_{SL}
5	
6	min T_{SDR}
7	
8	max T_{SDR}
9	
10	T_{QUI}
11	T_{SET}
12	T_{TR}
13	
14	
15	
16	G
17	HSA
18	max retry limit

Figure 9: Structure of "PBMasterParameters for a PBMasterSubmodule"

The parameters **should** be preset the PN/PROFIBUS Linking Device by manufacturer specific default values. The particular parameters are defined in IEC 61158-3 and 61158-4 Type 3 clauses as well in IEC 61158-5, chapter 8.2.6.2.2.3.

The manufacturer **may** transfer more parameters than specified above for his implementation, but has to accept if only the record specified above is sent. Possibly extensions of the implementation guide at hand will be realised by changing the Version fields of PBMasterParameters for a PBMasterSubmodule.

8.6.3 PROFIBUS Set Slave Address

The PROFIBUS Set_Slave_Add service shall be mapped to *PBSetSlaveAddressRecordData*. This data record **should** not be offered in the GSDML but only be used by the application program.
If the slave rejects the Set_Slave_Add the following negative confirmation should be used:

–	ErrorDecode	= PNIORW	
–	ErrorCode1	= 0xB2	(ErrorClass = 11 (Access error) / ErrorCode =02 (invalid slot/subslot))
		= 0xB5	(ErrorClass = 11 (Access error) / ErrorCode =05 (state conflict))
		= 0xB8	(ErrorClass = 11 (Access error) / ErrorCode =08 (invalid parameter))
–	ErrorCode2	= <*user specific*>	

Byte №	Description
0	Major Version = 0x00
1	Minor Version = 0x00
2	Old_Slave_Addr
3	New_slave_addr
4	Ident_Number
5	
6	No_Add_Chg
7	Rem_Slave_Data_Len
8	Rem_Slave_Data
…	
247	

Figure 10: Structure of "PBSetSlaveAddressRecordData"

Parameters: See [2].

8.6.4 PROFIBUS C2 Initiate Telegram Content

The PROFIBUS C2 Initiate Telegram content shall be mapped to PBSlaveInitiateRecordData. This data **should** not be offered in the GSDML.

Byte №	Description
0	Major Version = 0x00
1	Minor Version = 0x00
2	Send_Timeout
3	
4	Features_Supported
5	
6	Profile_Features_Supported
7	
8	Profile_Ident_Number
9	
10	Add_Addr_Param
…	
240	

Figure 11: Structure of "PBSlaveInitiateRecordData"

Parameters: See [2].

The PBSlaveInitiateRecordData contains the PROFIBUS Initiate.req telegram content excluding FunctionNum and three reserved bytes. It is left to the manufacturer to fill the structure with suitable default values.

The record exists for every slave so the initiate content can be adjusted individually for each slave. Whenever the Linking Device opens a C2 connection to a slave, the parameters in this record shall be used to initiate the acyclic connection.

8.7 PROFIBUS Slave Modules (*PBSlaveModules*)

8.7.1 Modelling of PROFIBUS Slaves

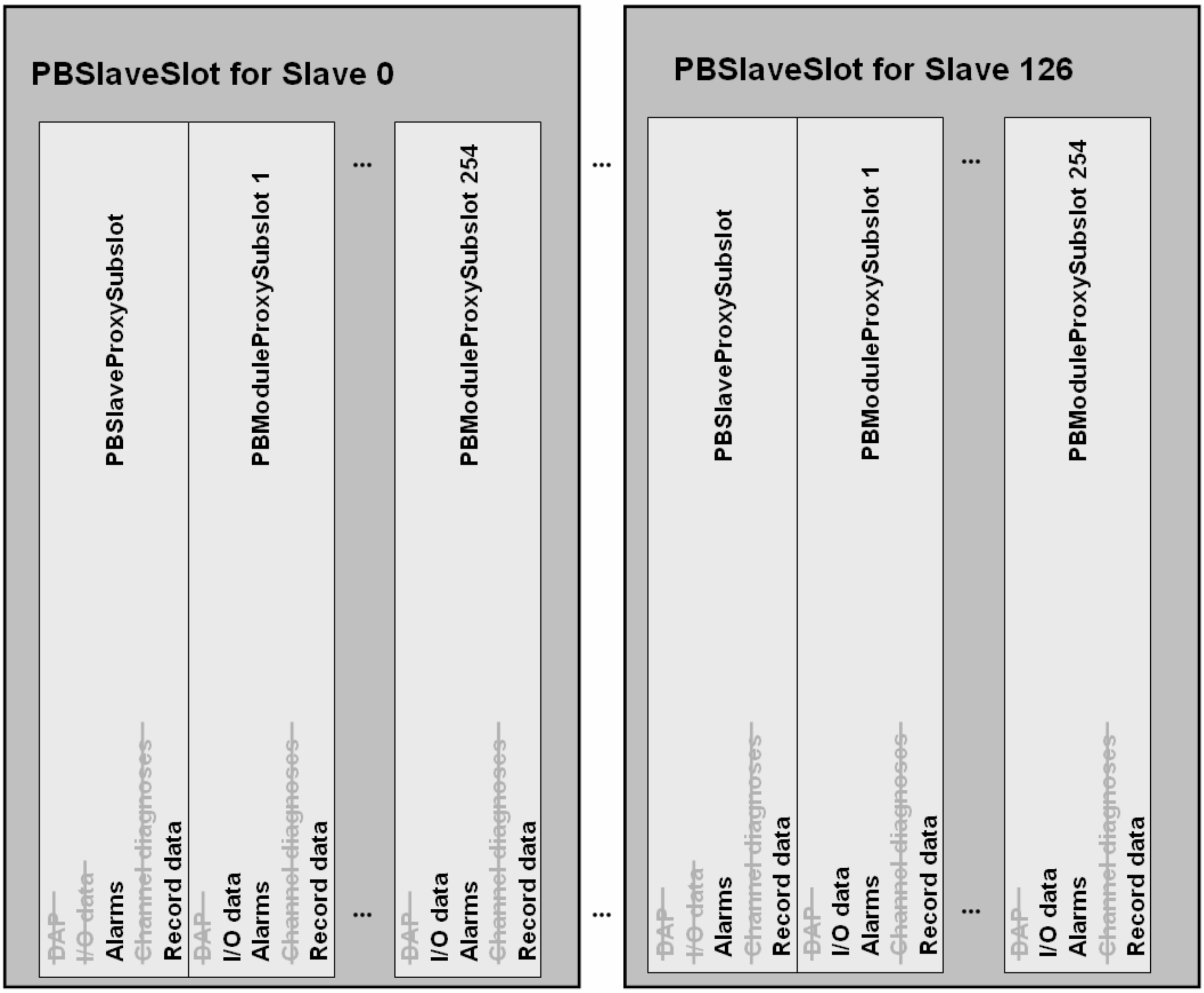


Figure 12: Modelling of the submodules of PROFIBUS slaves at the 1st Master System

The modules of each PROFIBUS slave **should** be restricted to 254 in maximum. Each *PBModuleProxySubmodule* **should** represent a slot of the physical PROFIBUS slave. The *PBSlaveProxySubmodule* represents the slave device itself.

The identification numbers for the *PBSlaveModule*, the *PBSlaveProxySubmodule* and *PBModuleProxySubmodule* are specified in the chapters 8.3 and 8.4.

The submodules of the *PBSlaveModule* **shall** support elements of following PROFINET-IO ASEs:

– I/O data	details see 8.7.2, 8.7.3
– Alarm	details see 8.7.4
– Data records	details see 8.7.6

8.7.2 Input Data and Status of PBModuleProxySubmodule

The Mapping Application transfers the I/O data of a submodule consistently from PROFIBUS to Ethernet or vice versa. Hence there is consistency of I/O data per submodule for the entire transmission line from the IO controller through the Linking Device to the PROFIBUS slave. The maximum size of a submodule is limited by the maximum size of a slot on the PROFIBUS and thus amounts to 128 bytes.

The IOPS of the inputs shows the availability of the module from the PROFIBUS master view. With IOPS = "Good", the last received input data of the PROFIBUS slave (current content of the transfer memory of the PROFIBUS master) **shall** be forwarded to the IO controller in addition.

The initial status of a module from the PROFIBUS master view **shall** be "not available" (corresponds to IOPS = "Bad").

The following events **shall** change the status of a module in the PROFIBUS master from "non available" to "available" and thus lead to an IOPS = "Good":

- Plug alarm received on PROFIBUS (only if DPV1)
- PROFIBUS stations enters data transfer phase (min. one DataExchange telegram was transmitted successfully to the PROFIBUS slave) and module exists.

The following events **shall** change the status of a module in the PROFIBUS master from "available" to "non-available" and thus lead to an IOPS = "Bad":

- Pull alarm received on PROFIBUS (only if DPV1)
- Station not in Data Exchange
- IO controller failure detected

The IOCS of the inputs **should** be ignored.

In PROFINET IO instead of "Substitute Values" the IOPS receives information about the validity of the data transmitted in the IO data object. With an IOPS = "Bad", the provider signals the failure of the IO data.

The IOPS and the transfer values **shall** be set according to Table 6.

Condition			**Reaction**	
PROFIBUS slave unavailable/ fault on transmission route	**Module of PROFIBUS slave pulled**	**Failure of PROFIBUS slave module**[1]	**IOPS**	**Transfer value**
False	False	False	Good	Original
False	False	True	Good	Original[2]
False	True	-	Bad	*Not defined*
True	-	-	Bad	*Not defined*

Table 6: Composing IOPS and transfer values of the inputs

The data sink is responsible for connecting the substitute values in case of IOPS being bad; in the case of inputs this is the host of the IO controller.

With the IOPS of the inputs the Linking Device signals the status of the PROFIBUS transmission line plus the status of the input module. Transmission line faults (failure of a PROFIBUS slave) **shall** lead to an IOPS = "Bad" because in this case no valid data are transmitted on the PROFIBUS.

[1] As an example a PROFIBUS module could have a wire failure that may be mapped to a PROFIBUS channel diagnosis. Hence the slave signals a diagnosis. If the ExtDiag bit is set this is reported as an alarm to the IO-Controller application. Nevertheless the received input data are transferred to the user application.

[2] In this case slave specific substitute value behaviour is applied. The Mapping Application forwards the received value.

With IOPS = “Good”, the Linking Device **shall** signal the successful reception of the input data on the PROFIBUS; this is possible in the operating status Clear as well as Operate of the PROFIBUS master.

8.7.3 Output Data and Status of PBModuleProxySubmodule

When IOPS = “Good”, the data received from the IO controller **shall** be adopted in the transfer memory of the PROFIBUS master.

When IOPS = “Bad”, the last value **shall** be held or a configured substitute value is entered in the transfer memory of the PROFIBUS master, depending on the substitute value strategy.(e.g. fix value zero)

With the IOCS of the outputs the Linking Device acknowledges the successful transmission of the output data by the IO controller into the transfer memory of the PROFIBUS master. Transmission of data on the PROFIBUS **shall** be still not assured however. This case applies if the station fails before the data was transmitted on PROFIBUS. In addition the IOCS of the outputs shows the current availability of the module from the PROFIBUS master view.

The initial status of a module from the PROFIBUS master view **shall** be “non-available” (corresponds to IOPS = “Bad”)

The following events **shall** change the status of a module in the PROFIBUS master from “non-available” to “available” and thus lead to an IOCS = “Good”:

- Plug alarm received on PROFIBUS (only if DPV1)
- PROFIBUS stations enters data transfer phase (min. one DataExchange telegram was transmitted successfully to the PROFIBUS slave) and module exists.

The following events **shall** change the status of a module in the PROFIBUS master from “available” to “non-available” and thus lead to an IOCS = “Bad”:

- Pull alarm received from PROFIBUS (only if DPV1)
- Station failure detected on the PROFIBUS
- IO controller failure detected

In the Clear operating status of the PROFIBUS master, the IOCS of the outputs may be “Good” (if module available), the IOCS of the outputs **shall** be set according to Table 7. The current output data of the IO controller **shall** be adopted in the transfer memory of the PROFIBUS master. The transfer values **shall** be determined according to Table 8.

Condition			**Reaction**
PROFIBUS slave unavailable/ fault on transmission route	**Module of PROFIBUS slave pulled**	**Failure of PROFIBUS slave module**[3]	**IOCS**
False	False	False	Good
False	False	True	Good
False	True	-	Bad
True	-	-	Bad

Table 7: Composing IOCS of the outputs

[3] As an example a PROFIBUS module could have a wire failure that may be mapped to a PROFIBUS channel diagnosis. Hence the slave signals diagnosis. If the ExtDiag bit is set this is reported as an alarm to the IO-Controller. Nevertheless the received output data are transferred to the slave.

Condition			**Reaction**
PROFIBUS slave unavailable/ fault on transmission route	**APDU Status.DataStatus. ProviderState**	**IOPS**	**Transfer value**
False	Run	Good	Original
False	Run	Bad	Substitute[4]
False	Stop	Good	Substitute = 0
False	Stop	Bad	Substitute = 0
True	-	-	-

Table 8: Composing transfer values of the outputs

As the IOPS of the outputs cannot be transmitted on the PROFIBUS, the Mapping Application has to adopt substitute value connection as proxy for the actual data sink. This means that when IOPS = “Bad”, the substitute values of the PROFIBUS master (depending to the implementation of the PROFIBUS master) are adopted in the transfer memory of the PROFIBUS master. In the next PROFIBUS cycle (and no Clear is set on the PROFIBUS) they are then transmitted to the PROFIBUS slave and hence to the actual data sink.

In the operating status Clear of the PROFIBUS master and during faults on the PROFIBUS (PROFIBUS master failure) the PROFIBUS slave connects its own substitute values to the I/Os. These values can e.g. be selected by default by hardware properties of the module.

For a uniform substitute value behavior the substitute values in the PROFIBUS master (depending to the PROFIBUS master implementation) and in the PROFIBUS slave have to be identical. Changing the substitute value behavior in runtime is the user’s responsibility. The PROFIBUS master does not detect this change; the substitute value behavior between module and PROFIBUS master thus becomes inconsistent.

[4] In this case the substitute value/ strategy has to be chosen by the linking device

8.7.4 Alarms on PROFINET

The table below specifies alarm types and their special usage for the PROFIBUS integration that **shall** be provided by PROFIBUS slave proxy modules. For information under which conditions these alarms have to be initiated, refer to chapter 8.9.3.

PROFINET Alarm Type	Subslots	Usage
Diagnosis	PBSlaveProxySublslot and PBModuleProxySublsots	PROFIBUS Diagnosis alarms of the PROFIBUS slaves are mapped to PROFINET IO Diagnosis alarms.
	PBSlaveProxySublslot	Signals the event of a PROFIBUS diagnosis. For this purpose special values for PN AlarmSpecifier are defined, see 8.7.4.4. The explicit mapping is described in chapter 8.9.3
Diagnosis disappears	PBSlaveProxySublslot	Signals a disappearing diagnosis event of the PROFIBUS slave. The according AlarmSpecifier is defined in 8.7.4.4 Reset of Ext_Diag Flag inside the PROFIBUS Diagnosis
Status	PBSlaveProxySublslot and PBModuleProxySubslots	Signals the event of a PROFIBUS status alarm. For this purpose special values for AlarmSpecifier are defined, see 8.7.4.4. The explicit mapping is described in 8.9.3
Update	PBSlaveProxySublslot and PBModuleProxySubslots	Signals the event of a PROFIBUS update alarm. For this purpose special values for AlarmSpecifier are defined, see 8.7.4.4. The explicit mapping is described in 8.9.3
Process	PBModuleProxySubslots	Process alarms of the PROFIBUS slaves are mapped to PROFINET IO process alarms.
Pull	PBSlaveProxySublslot and PBModuleProxySubslots	A PROFIBUS Slave proxy module signals the failure of a slave. The alarm affects all subslots of a module though signalled always at PBSlaveProxyModule.
Plug	PBSlaveProxySublslot and PBModuleProxySubslots	A PROFIBUS Slave proxy module signals the return of a configured slave (and so a new need for parameterisation).
Controlled by supervisor	PBSlaveProxySublslot and PBModuleProxySubslots	This alarm **shall** be implemented unless the Mapping Application supports no Supervisor AR
Released	PBSlaveProxySublslot and PBModuleProxySubslots	This alarm **shall** be implemented unless the Mapping Application supports no Supervisor AR
Manufacturer Specific	PBSlaveProxySublslot and PBModuleProxySubslots	All alarms from 0x20 to 0x 7E **shall** be passed to the according manufacturer specific PROFIBUS alarm.

Table 9: Alarms with special usages for the mapping of PROFIBUS slaves in PBSlaveModules

8.7.4.1 Mapping Of PROFIBUS Diagnoses To PROFINET Alarms

The following Figure 13 shows the implementation of the PROFIBUS diagnosis data in a PROFINET IO alarm request block.

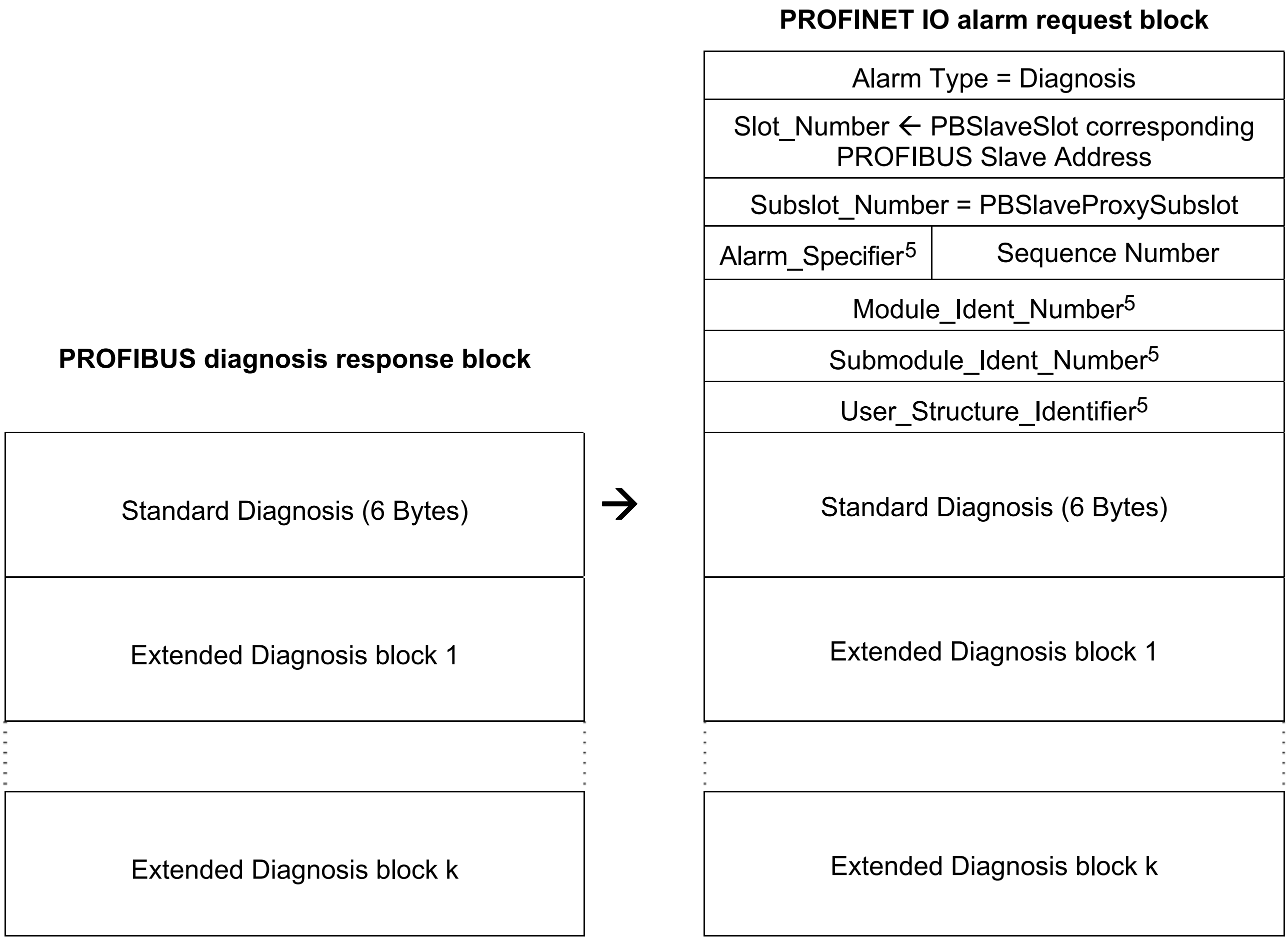

PROFINET IO alarm request block
Alarm Type = Diagnosis
Slot_Number ← PBSlaveSlot corresponding PROFIBUS Slave Address
Subslot_Number = PBSlaveProxySubslot
Alarm_Specifier[5]
Sequence Number
Module_Ident_Number[5]
Submodule_Ident_Number[5]
User_Structure_Identifier[5]
PROFIBUS diagnosis response block
Standard Diagnosis (6 Bytes)
→
Standard Diagnosis (6 Bytes)
Extended Diagnosis block 1
Extended Diagnosis block 1
Extended Diagnosis block k
Extended Diagnosis block k

Figure 13: Mapping of the PROFIBUS V0 diagnosis to PROFINET IO alarm request block

The following PROFINET alarm behaviour for the corresponding PBSlaveModule should be implemented:

1. When the Ext_Diag Bit in the PROFIBUS Diagnosis-RES-PDU is not set (0), no PROFINET IO alarm should be generated.
2. When the Ext_Diag Bit in the PROFIBUS Diagnosis-RES-PDU is set (1), a PROFINET IO diagnosis alarm should be generated for *every* new PROFIBUS diagnosis.
3. For a transition of the Ext_Diag bit in the PROFIBUS Diagnosis-RES-PDU from 1 to 0 a PROFINET IO diagnosis disappears alarm should be generated.

The evaluation of the PROFIBUS diagnosis data and the triggering of PROFINET diagnosis alarms is handled by the Mapping Application.

[5] To be supplemented by the Mapping Application according to the chapters 8.7.4.3 and 8.9.3

8.7.4.2 Mapping Of DPV1 Alarms to PROFINET Alarms

The following Figure 14 shows the implementation of the PROFIBUS alarm request block in a PROFINET IO alarm request block.

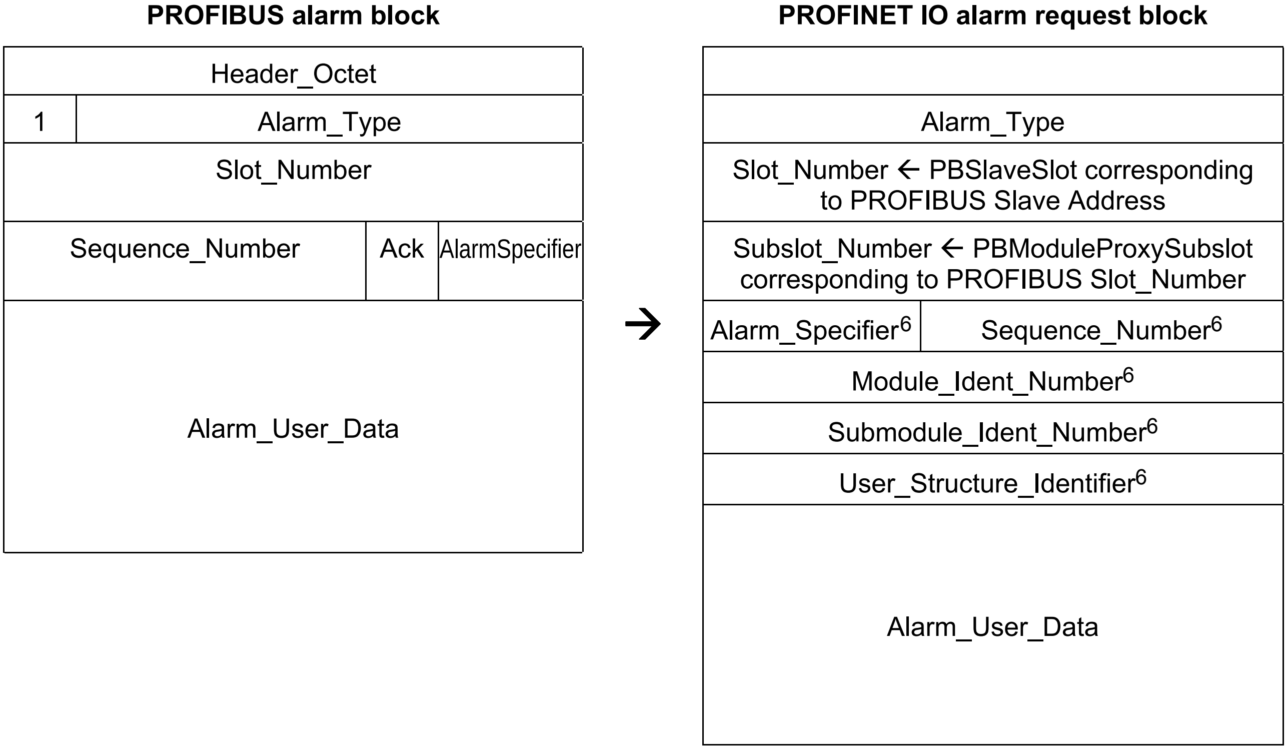

PROFIBUS alarm block
Header_Octet
1
Alarm_Type
Slot_Number
Sequence_Number
Ack
AlarmSpecifier
Alarm_User_Data
→
PROFINET IO alarm request block
Alarm_Type
Slot_Number ← PBSlaveSlot corresponding to PROFIBUS Slave Address
Subslot_Number ← PBModuleProxySubslot corresponding to PROFIBUS Slot_Number
Alarm_Specifier[6]
Sequence_Number[6]
Module_Ident_Number[6]
Submodule_Ident_Number[6]
User_Structure_Identifier[6]
Alarm_User_Data

Figure 14: Mapping of the PROFIBUS alarm request block to PROFINET alarm request block

For each PROFIBUS alarm of the PROFIBUS slave an analogous PROFINET IO alarm **should** be created by the Mapping Application.

8.7.4.3 UserStructureIdentifier

The following division of the "UserStructureIdentifier" number range is specified for PROFIBUS.

Value	Name	Meaning
0x0000 – 0x2000		Reserved for vendor-specific use
0x2001 – 0x4000	PROFIBUS_SLAVE_DIAG	PROFIBUS slave diagnosis telegram (see [2]). The Value(s) used is manufacturer specific. An application using this value to interpret the data structure shall accept all values within this range.
0x4001 – 0x6000	PROFIBUS_DPV1_ALARM	PROFIBUS DPV1 additional alarm info (see [2]). The Value(s) used is manufacturer specific. An application using this value to interpret the data structure shall accept all values within this range.
0x6001 – 0x7FFF	-	Reserved for profile-specific use within the PROFIBUS profile

Table 10: UserStructureIdentifier for PROFIBUS mapping

[6] To be supplemented by the Mapping Application according to the following chapters

8.7.4.4 AlarmSpecifier

The Mapping Application creates the AlarmSpecifier as follows:

AlarmSpecifier. ManufacturerSpecificDiagnosis

Shall be TRUE.

AlarmSpecifier.SubmoduleDiagnosisState

Upon diagnosis alarms the Mapping Application uses the PROFIBUS AlarmSpecifier to form the PROFINET IO SubmoduleDiagnosisState as follows:

PROFIBUS AlarmSepcifier (see [2])	PROFINET IO AlarmSpecifier. SubmoduleDiagnosisState (see [1])
00 : unspecified	NO_DIAG
01: error coming, slot faulted	DIAG
10: error going, slot error-free	NO_DIAG
11: error going, slot faulted	DIAG

Table 11: Definition for AlarmSpecifier for PROFIBUS mapping

8.7.5 Channel Diagnosis

No Channel Diagnoses are generated for the PROFIBUS slave proxy at subslot, slot or device level. The diagnosis data of a PROFIBUS slave are stored as "ManufacturerSpecificDiagnosisData" on the master proxy. The standardised indices for ChannelDiagnosisData are answered at least on all levels with a length of 0 byte. The manufacturer **may** implement these Channel Diagnoses as an extension.

8.7.6 Parameterisation Record Data

8.7.6.1 PROFIBUS Device Level SET_PRM Data (PBSlavePRMRecordData)

The PN/PROFIBUS Linking Device **shall** implement the start-up data record as defined in Figure 15 for all *PBSlaveProxySubmodules*. This data record is offered in the GSDML for configuration and parameterisation.

Byte №	Description
0	Major Version = 0x00
1	Minor Version = 0x00
2	Station_Status
3	WD_Fact_1
4	WD_Fact_2
5	Min_T$_{SDR}$
6	Ident_Number
7	
8	Group_Ident
9	Length of DPV1_Status
10	DPV1_Status_1
11	DPV1_Status_2
12	DPV1_Status_3
13	Length of XXX_Prm_Data
14	XXX_Prm_Data
..	
Max 250	

Figure 15: Structure of the start up data record “PBSlavePRMRecordData”

Parameters:

Station_Status	see [2]
Length of DPV1_Status	0 (if not DPV1), 3 (if DPV1)
DPV1_Status_1	see [2]
DPV1_Status_2	see [2]
DPV1_Status_3	see [2]
Length of XXX_Prm_Data	0 .. 223

XXX_Prm_Data	if PRM_Structure in DPV1_Status_3 = 0:
	List of User_Prm_Data_Element
	if PRM_Structure in DPV1_Status_3 = 1:
	List of Structured_Prm_Data

On the basis of the record data PBSlavePRMRecordData and PBModulePrmRecordData a Set_Prm-REQ-PDU or a Set_Ext_Prm-REQ-PDU has to be built by the Mapping Application as described below in chapter 8.7.6.2.

8.7.6.2 PROFIBUS Slot Level USER_PRM Data (PBModulePRMRecordData)

Additionally to the device PRM values defined in 8.7.6.1 module specific User_Prm_Data are to be transferred. The so configured module specific blocks **shall** be compiled to a PROFIBUS Set_Prm-REQ-PDU with following rules:

- PRM_Structure in DPV1_Status_3 = 0: XXX_Prm_Data is not structured in blocks. It **shall** be concatenated and sent to the slave as a normal User_Prm_Data block. The data ordering shall be slave specific data first followed by module specific data ascending with the PROFIBUS slot number as described in Figure 17 below.
- PRM_Structure in DPV1_Status_3 = 1: XXX_Prm_Data consists of Structured_Prm_Data blocks. All blocks of the same type **shall** be concatenated as defined below. The compiled PRM telegram shall be sent to the slave as Structured_Prm_Data. The data ordering within each of the compiled blocks is starting with the slave specific data followed by the module specific blocks ascending with the PROFIBUS Slot_Number as defined in Figure 18 below.
- The total Set_Prm length of 244 bytes **shall** not be exceeded.
- Only one single Structure_Type and Slot_Number for a joker block[7] **shall** be transferred to a slave.
- The Set_Ext_Prm-REQ (and thus joker blocks to the Set_Ext_Prm-REQ) **may** not be supported.

Byte №	Description
0	Major Version = 0x00
1	Minor Version = 0x00
2	Length of PROFIBUS Module Specific XXX_Prm_Data
3	XXX_Prm_Data
..	
max 239	

Figure 16: Structure of the start up data record "PBModulePRMRecordData"

Parameters:

Length of Module Specific XXX_Prm_Data	0 (if no module specific PRM data is configured), 1.. 223 (otherwise)[8]
XXX_Prm_Data	see [2].

The block ordering in Figure 17 and Figure 18 apply for building the PROFIBUS Prm_Data:

[7] A joker block is to be identified by Structure_Length = 255.

[8] Additionally the total length of the compiled PROFIBUS Set_Prm telegram must not be exceeded

Ordering rank	Block/ content
0	PBSlavePRMRecordData. XXX_Prm_Data *<if present>*
1	PBModulePrmRecordData. XXX_Prm_Data *<of module 1, if present>*
2	PBModulePrmRecordData. XXX_Prm_Data *<of module 2, if present>*
254	PBModulePrmRecordData. XXX_Prm_Data *<of module 254, if present>*

Figure 17: Format of User_Prm_Data if not DPV1

In case of DPV1 the structure of XXX_Prm_Data is an aggregation of elements with type Structured_Prm_Data. These elements are composed according to Figure 18 below. The composition of the Structured_Prm_Data_Block [Structure_Type, Slot_Number] itself is defined in Figure 19.

Ordering rank	XXX_Prm_ Data. Structure_ Type (= σ)	XXX_Prm_ Data. Slot_Num ber (= τ)	Block/ content
0	-	-	PBSlavePRMRecordData. DPV1_Status_1
			PBSlavePRMRecordData. DPV1_Status_2
			PBSlavePRMRecordData. DPV1_Status_3
3	2	0	Structured_Prm_Data_Block [2, 0] *<if block is present & no joker >*
		1	Structured_Prm_Data_Block [2, 1] *<if block is present & no joker >*
		254	Structured_Prm_Data_Block [2, 254] *<if block is present & no joker >*
4	3	0	Structured_Prm_Data_Block [3, 0] *<if block is present & no joker >*
		1	Structured_Prm_Data_Block [3, 1] *<if block is present & no joker >*
		254	Structured_Prm_Data_Block [3, 254] *<if block is present & no joker >*
5	4	0	Structured_Prm_Data_Block [4, 0] *<if block is present & no joker >*
		1	Structured_Prm_Data_Block [4, 1] *<if block is present & no joker >*
		254	Structured_Prm_Data_Block [4, 254] *<if block is present & no joker >*
6	7	0	Structured_Prm_Data_Block [7, 0] *<if block is present & no joker >*
		1	Structured_Prm_Data_Block [7, 1] *<if block is present & no joker >*
		254	Structured_Prm_Data_Block [7, 254] *<if block is present & no joker >*
7	32	0	Structured_Prm_Data_Block [32, 0] *<if block is present & no joker >*
		1	Structured_Prm_Data_Block [32, 1] *<if block is present & no joker >*
		254	Structured_Prm_Data_Block [32, 254] *<if block is present & no joker >*
135	129	0	Structured_Prm_Data_Block [129, 0] *<if block is present & no joker >*
		1	Structured_Prm_Data_Block [129, 1] *<if block is present & no joker >*
		254	Structured_Prm_Data_Block [129, 254] *<if block is present & no joker >*
136	32 … 129	0 .. 254	Structured_Prm_Data_Block [τ, σ] *<if joker block present>*

Figure 18: Format of User_Prm_Data if DPV1

In order to compose a Structured_Prm_Data_Block of Structure_Type τ and PROFINET Slot_Number σ the Mapping Application has to reassemble them from the single blocks of Structured_Prm_Data in both

PBSlavePRMRecordData. XXX_Prm_Data and PBModulePrmRecordData. XXX_Prm_Data with identical Structrue_Type τ according to Figure 19.

Parameteter in Structured_ Prm_Data_Block	Source for a PROFIBUS slot s
Structure_Length	Σ_σ XXX_Prm_Data. Structured_Prm_Data [τ, σ]. Structure_Length *<if block is no joker & XXX_Prm_Data. Structured_Prm_Data [τ, σ]. Structure_Type = s>, 255 <otherwise>*
Structure_Type	τ
Slot_Number	XXX_Prm_Data. Structured_Prm_Data [τ, σ]. Slot_Number = s
reserved	0
User_Prm_Data	XXX_Prm_Data. Structured_Prm_Data [τ, 2]. User_Prm_Data_Element *<if present>*
	XXX_Prm_Data. Structured_Prm_Data [τ, 3]. User_Prm_Data_Element *<if present>*
	XXX_Prm_Data. Structured_Prm_Data [τ, 129]. User_Prm_Data_Element *<if present>*

Figure 19: Assembly of Structured_Prm_Data_Block

8.7.6.3 PROFIBUS CHK_CFG Data (PBConfigRecordData)

PN/PROFIBUS Linking Device **shall** implement the start-up data record defined in Figure 20 for all PBModuleProxySubmodules. This data record is offered in the GSDML for configuration and parameterisation for every PBModuleProxySubmodule.

Byte №	Description
0	Major Version = 0x00
1	Minor Version = 0x00
2	AKF/ SKF (1st Byte)
3	SKF (2nd to max 16th Byte)
..	
Max 17	

Figure 20: Structure of the start up data record “Config data”

Parameters:

AKF	see [2]
SKF	see [2].

8.7.6.4 PROFIBUS GET_CONFIG Service

The PROFIBUS service Get_Cfg **shall** be mapped to a read of Record data. This data record **should** not be offered in the GSDML but only be used by the application program. The structure of the data record read response **shall** be defined by “Get_Cfg-RES-PDU” according to [2].

8.7.6.5 PROFIBUS RD_INPUT

The PROFIBUS READ_INPUT service **shall** be mapped to *PBReadInputRecordData* in *PBSlaveProxy-Submodule*. The data record read response **shall** keep the plain data of the PROFIBUS service RD_Input-RES-PDU. This data record **shall** not be accessible by a data record write request.

8.7.6.6 PROFIBUS RD_OUTPUT

The PROFIBUS READ_OUTPUT service **shall** be mapped to *PBReadOutputRecordData* in *PBSlaveProxySubmodule*. The data record read response **shall** keep the plain data of the PROFIBUS service RD_Output-RES-PDU. This data record **shall** not be accessible by a data record write request.

8.7.7 PBSlaveRecordData

The record data objects of the IO device instances can be read or written by the IO controller as well as by the IO supervisor. Record data that are addressed to an unknown submodule of the Linking Device result in rejecting the job is with the error "invalid slot/subslot" (0xB2).

8.7.7.1 Mapping of PBSlaveRecordData

Record data objects with an index from 0x0000 – 0x00FF **shall** be directly read or written from/to the PROFIBUS slave. The PROFINET IO record data objects **should** be mapped on

- the PROFIBUS MS1 channel if the accessing PROFINET AR was an IO-AR or a AR with DeviceAccess = false.
- the PROFIBUS MS2 channel[9] if the accessing PROFINET AR was a Supervisor-AR with or without DeviceAccess

The PN/PROFIBUS Mapping Application **shall** forward all jobs with an index: 0x0000 ... 0x00FF (independently of the mapped PROFIBUS slave type) to the corresponding PROFIBUS channel. No proxy data records slave **should** be created in the PROFIBUS master.

The error messages of the PROFIBUS MS1/MS2 channel **should** be mapped as follows in PROFINET IO:

- PROFIBUS "Error_Decode" corresponds to PROFINET IO "ErrorDecode" (value: 0x80 = PNIORW)
- PROFIBUS "Error_Code1" can be adopted 1 : 1 as PROFINET IO "ErrorCode1" as the PROFIBUS "Error_Code1" was adopted in PROFINET IO.
- PROFIBUS "Error_Code2" can be adopted 1 : 1 as PROFINET IO "ErrorCode2" because PROFIBUS as well as "Error_Code2" in PROFINET IO was identified as vendor-specific.
- If a record is to be written with a write length > 240 bytes, "WriteLengthError" (ErrorCode1 = 0xB1) is transmitted by the PN/PROFIBUS Linking Device
- If a record is to be read with a read length > 240 bytes, a read job with 240 bytes is transmitted on the PROFIBUS by the PN/PROFIBUS Linking Device
- If there is no PROFIBUS response (RS), the PROFINET IO ErrorCode1/ ErrorCode2 should be mapped to resource unavailable.

8.7.7.2 Non-Implemented Record Data

Reading non-implemented indices (profile or manufacturer specific entries) **should** be rejected with the following error numbers:

- ErrorDecode = 0x80 (PNIORW)
- ErrorCode1 = 0xB0 (ErrorClass = 11 (access error) / ErrorCode = 0 (invalid index))
- ErrorCode2 = *<user specific>*

8.7.8 Standardised Record Data

The description of these individual data records is specified in [1]. Only PN/PROFIBUS elements specific for the PN/PROFIBUS Linking Device are considered in the following.

8.7.8.1 Manufacturer Specific Diagnosis Data

The current PROFIBUS slave diagnosis data are made available as ManufacturerSpecificDiagnosisData according to [1] (index 0xF00B) on slave proxy modules (subslot 0x1000) of each proxy IO device. The content of the data record corresponds to the currently available data of the PROFIBUS slave diagnosis telegram.
The UserStructureIdentifier is PROFIBUS_SLAVE_DIAG (see chapter 8.7.4.3).

8.7.8.2I&M Record Data

I&M Data, if the PROFIBUS slave supports it, **may** be mapped to the PROFINET I&M indices of the PBSlaveProxySubmodule and PBModuleProxySubmodules.

8.8 Device Description

8.8.1 Channel Diagnoses

No Channel Diagnosis is used at this specification.

[9] Because definition of the IO Supervisor is in progress at the moment, this definition might be extended by the ongoing definition.

8.8.2 Parameterisation Data Records

Parameterisation data records are defined in detail by Figure 9, Figure 15 , Figure 16 and Figure 20.

8.8.3 GSDML Properties

8.8.3.1 Device Access Point

The Linking Device **should** offer different DAPs in order to use it as a PROFINET V1 or a PROFINET V2 device. The details are worked out in chapter 9.1.

8.8.3.2 Physical Slots

A definition of as slot list together with the attribute “PhysicalSlots” causes undefined slots not to be shown in the engineering surface. So in a GSDML description **should** be provided with an appropriate slot list. This results in a clearly arranged sight in the engineering where only the usable slots will be seen.

8.8.3.3 Usable Modules

The PBMasterSubmodule **shall** be described in the GSDML as ’VirtualSubmoduleList’.

8.9 Dynamic Behaviour

8.9.1 PROFIBUS Master Operation Modes

The APDUStatus.DataStatus.ProviderState: Stop/ Run of the IO AR is mapped on the Clear-signal of the PROFIBUS master. Therewith the PROFIBUS ‘Clear’ operation can be controlled by the user of the PROFINET IO Controller. One PROFIBUS Master System **shall** be controlled by one IO AR solely.

In the beginning of the start-up of the linking device a save output of the PROFIBUS slaves **shall** be forced by executing a MasterStateClear. Independent to this the received input data are transferred. At the end of the start-up phase this behaviour **shall** be replaced by assuming the current provider state of the PROFIBUS Master System.

During the distribution of ‘Clear’ on the PROFIBUS, ZERO[10] data are transmitted to the PROFIBUS slave for all outputs of the PROFIBUS Master System. This leads in the PROFIBUS slave to "disconnection of process outputs" or to connection of a predefined substitute value. The transmission of the input data does not depend on the Clear-signal.

The PROFIBUS masters of a Linking Device provide Global Control on ‘Clear’ if the APDU provider status of the IO AR is stated in Table 12.

Condition	Reaction
AR ASE: **ProviderState of IO AR**	**Behaviour of Global Control**
Run	No clear
Stop	Clear

Table 12: Interconnection between APDU status of output CR and PROFIBUS Global Control

8.9.2 Expected Identification and Real Identification

When a PROFINET IO-Controller takes control over submodules establishing the IO-CR, there is to be decided on the Mapping Application whether the Expected Identification fits to the Real Identification. Particularly following parameters **shall** be checked:

- Module Ident Number
- Submodule Ident Number
- Availability of the module

The definition, which Module/ Submodule Ident Number of a *PBSlaveModule* **shall** be accepted for a real PROFIBUS slave, is given in chapter 8.9.2.2.

A *PBSlaveModule* is available

[10] A data telegram filled with 0 or without content (FailSafe).

- for an IO Controller if none of its submodules are controlled by IO Supervisor (via a Supervisor AR) and not controlled by another IO Controller.
- for an IO Supervisor if no controller has taken any of its submodules unless the attribute 'Takeover Allowed' of the taken submodule is set to 'ALLOWED'.

8.9.2.1 Rules For DeviceModule And PBMasterModules

The ModuleState in the ModuleDiffBlock in the IODConnectResponse **should** be set according to Table 13.

Preconditions	**ModuleState**
Supported by Linking Device	
False	No module
True	Proper module

Table 13: Parameterisation of PBMasterModules - ModuleDiffBlock.ModuleState

The *PBMasterSubmodules* **shall** not be taken over by an IO Supervisor. Therefore, the Parameter SubmoduleState.AddInfo for these submodules **should** be set in the ModuleDiffBlock of the IODConnectResponse according to Table 14 (FormatIndicator required to be '1').

Rule №	**Preconditions**	**SubmoduleState**				
		Format Indicator	**AddInfo**	**DiagInfo**	**ARInfo**	**IdentInfo**
1	<None>	1	Takeover is not allowed	No Diagnosis Data available	Application Ready Pending	OK

Table 14: Parameterisation of PBMasterSubmodules - ModuleDiffBlock.SubmoduleState

8.9.2.2 Rules For PBSlaveModules

The comparison of the Expected Identification and Real Identification data is not possible for all module types at AR connect time. The complete description of the I/O data is transferred afterwards as record data (see chapter 8.7.6.1). So, always the ModuleDiffBlock returned with the Connect response **should** keep dummy entries for the PBSlaveModules. The real entries are returned with the ModuleDiffBlock in the ApplicationReady.ind.

Table 15 defines, specific for any PBMasterModule, how the correct Module state for the ModuleDiffBlock (and other context class services) **should** be determined.

Preconditions		**ModuleState**
Comparison of lifelist/ PROFIBUS Ident№ and ModuleID/ SubmoduleIDs	**Life list**	
---	0	No module
Difference Bit 15..0 ModuleID AND in PROFIBUS Ident № of Profibus slave	1	Wrong module
Differences in submodules	1	Proper module

Table 15: Parametrisation of slave proxy modules – ModuleDiffBlock.ModuleState

The Module State "Substitute" is not used.

The tables below define which SubmoduleState **should** be returned in the ModuleDiffBlock (and other context class services). The standard handling **should** be supported (defined by rules 3 to 6), the supervisor handling **may** be supported and in this case the rules 7 to 8 **should** be used.

Rule №	Preconditions	SubmoduleState				
		Format Indicator	AddInfo	DiagInfo	ARInfo	IdentInfo
2	<None>	1	*<Manufacturer specific>*	No Diagnosis Data available	Application Ready Pending	OK

Table 16: Parameterisation of PBModuleProxySubmodules - ModuleDiffBlock.SubmoduleState for Connect Response

Rule №	Preconditions				SubmoduleState				
	Life list	Prm Fault	Cfg Fault	Locked by	Format Indicator	AddInfo	DiagInfo	ARInfo	IdentInfo
3	0				---	---	---	---	---
4	1	0	0	none	1	<do not care>	No Diagnosis Data available	Own	OK
5	1	0	1	none	1	<do not care>	No Diagnosis Data available	Application Ready Pending	Substitute
6	1	1	-	none	1	<do not care>	No Diagnosis Data available	Application Ready Pending	Substitute
7	1	-	-	Supervisor	1	None	No Diagnosis Data available	Application Ready Pending	Locked By IO Supervisor
8	1	-	-	Controller	1	Takeover is not allowed	No Diagnosis Data available	Application Ready Pending	Locked By IO Controller

Table 17: Parameterisation of PBModuleProxySubmodules - ModuleDiffBlock.SubmoduleState for Application Ready

8.9.3 Alarms Rules

Independent of other definitions any change of an input IOXS from ‘bad’ to ‘good’ **shall** go along with an alarm ‘Return of submodule’:

Action	*Reaction*
Change of input IOPS	**Alarm on entering configuration mode**
Bad → good	Return of Submodule
Good → bad, no change	-

Table 18: Dependency of alarms types on master mode changes and a slave proxy’s input IOPS

Following rules **shall** be applied for diagnosis events:

Action	*Reaction*
PROFIBUS Diagnosis-RES-PDU	**PROFINET Alarm_Notification**

New diagnosis with Station_status_1.Diag.Ext_diag=*true* on slave with address α	CREP = <*current*>
	Alarm_Type = *Diagnosis*
	Slot_Number = PBSlaveSlot corresponding to PROFIBUS slave address α
	Subslot_Number = PBSlaveProxySubslot
	AlarmSpecifier according to 8.7.4.4
	Sequence_Number = <*non ambiguous number*>
	Module_Ident_Number = <*of module α*>
	Submodule_Ident_Number = <*of PBSlaveProxySubmodule*>
	Alarm_User_Data_Structure_Identifier according to 8.7.4.4
	Alarm_User_Data = Diagnosis_User_Data of received PROFIBUS Diagnosis-RES-PDU

New diagnosis with Station_status_1.Diag.Ext_diag=*false* on slave with address α	No reaction

Station_status_1.Diag.Ext_diag *true* → *false* transition (previous diagnosis: Ext_diag=true, new diagnosis Ext_diag=false) on slave with address α	CREP = <*current*>
	Alarm_Type = *Diagnosis disappears*
	Slot_Number = PBSlaveSlot corresponding to PROFIBUS slave address α
	Subslot_Number = PBSlaveProxySubslot
	AlarmSpecifier according to 8.7.4.4
	Sequence_Number = <*non ambiguous number*>
	Module_Ident_Number = <*of module α*>
	Submodule_Ident_Number = <*of PBSlaveProxySubmodule*>
	Alarm_User_Data_Structure_Identifier according to 8.7.4.3
	Alarm_User_Data = Empty

Table 19: Dependency between PROFIBUS diagnoses and PROFINET diagnosis alarms

If PROFIBUS diagnosis indicates a DPV1 alarm to the Mapping Application the rules in Table 20 have to be applied in addition. The acknowledge mapping is described in Table 21.

PROFIBUS action		*PROFINET reaction*
Diagnosis-RES-PDU. Alarm. Header_Octet. Selection	**Diagnosis-RES-PDU. Alarm. Alarm_Type**	**Alarm_Notification** with AlarmSpecifier according to 8.7.4.4

00_{bin}	-	Diagnosis on according slave proxy module:
		CREP = <*current*>
		Slot_Number = PBSlaveSlot corresponding to PROFIBUS slave address
		Subslot_Number = PBModuleProxySubslot corresponding to PROFIBUS Slot_Number
		Sequence_Number = Sequence_Number
		Module_Ident_Number = <*of current module*>
		Submodule_Ident_Number = <*of current submodule*>
		Alarm_User_Data = PROFIBUS Diagnosis_User_Data

	Diagnostic_Alarm	Alarm_Type = *Diagnosis*
	Process_Alarm	Alarm_Type = *Process*
	Pull_Alarm	Alarm_Type = *Pull*
	Plug_Alarm	Alarm_Type = *Plug*
	Status_Alarm	Alarm_Type = *Status*
	Update_Alarm	Alarm_Type = *Update*
	manufacturer-specific (32-126)	Alarm_Type = *Manufacturer Specific* (i.e. 32→ 32, 33→ 33, …, 126→126)

Table 20: Mapping of DPV1 alarms to PROFINET alarms

PROFINET action		*PROFIBUS reaction*
Alarm Ack. req	**Alarm Ack.req. Alarm_Type**	**Alarm_Ack-REQ-PDU**
On subslot number σ of a PBSlaveProxySubslot or PBModuleProxySubslot	-	Slot_Number = PROFIBUS slot number corresponding to PROFINET PBModuleProxySubslot number σ Alarm_Specifier according to [2]
	Diagnosis	Alarm_Type = Diagnostic_Alarm
	Process	Alarm_Type = Process_Alarm
	Pull	Alarm_Type = Pull_Alarm
	Plug	Alarm_Type = Plug_Alarm
	Status	Alarm_Type = Status_Alarm
	Update	Alarm_Type = Update_Alarm
	Manufacturer Specific	Alarm_Type = manufacturer-specific (i.e. 32→ 32, 33→ 33, …, 126→126)

Table 21: Mapping of alarm acknowledge from PROFINET to PROFIBUS

9 Implementation Hints

9.1 Modelling of The DAP

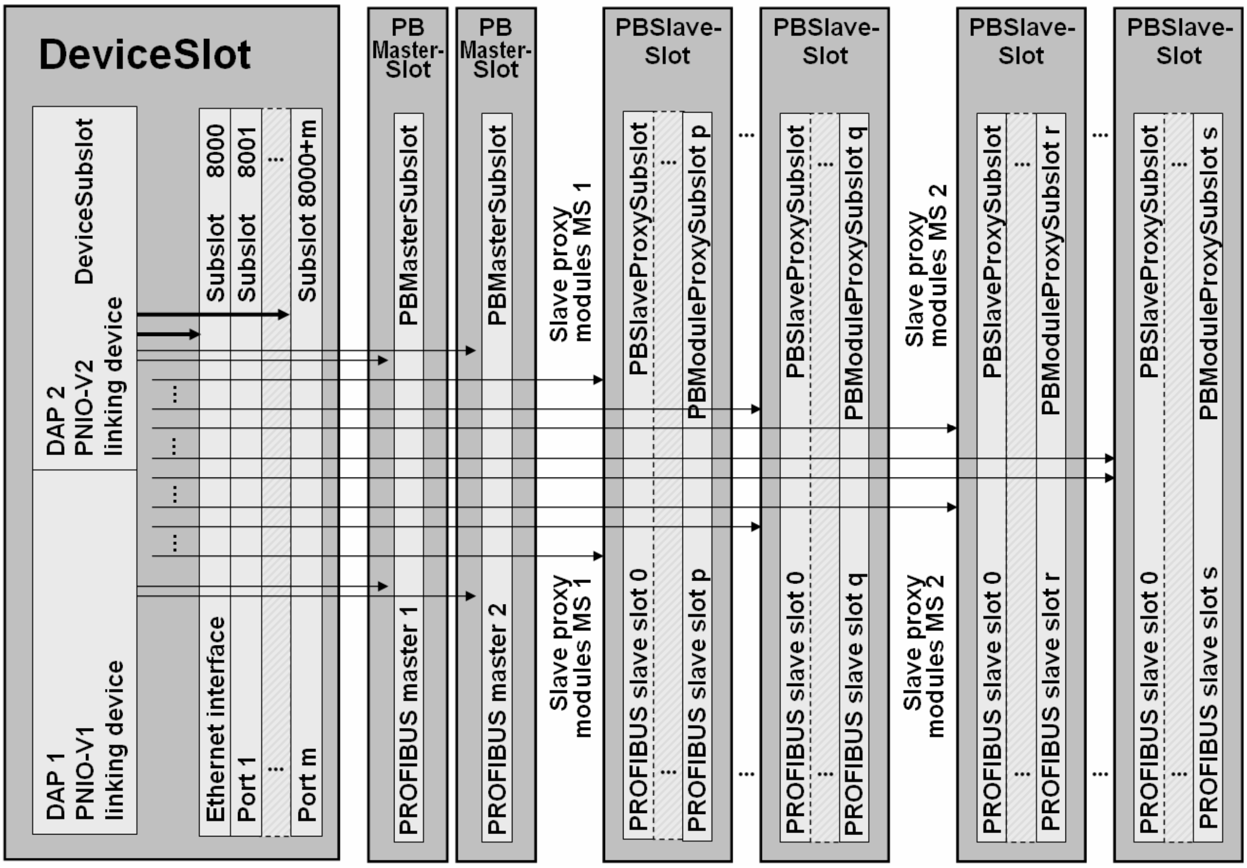


Figure 21: Modelling of DAPs

To avoid the formation of multiple GSDMLs addicted to the PROFINET IO controller version the user **should** be offered different devices with different DAPs for a Linking Device in one single GSDML. Each DAP **should** represent a device suitable for an according version of a PROFINET IO Controller. As an example, see Figure 21. In Figure 21 DAP 1 offers a PN/PROFIBUS Linking Device with two PROFIBUS Master System referring to the PROFINET specification V1 (no port diagnosis), whereas DAP 2 references a PROFINET IO Device for V2 controllers and so refers in addition to the submodules 0x8000 and higher (bold arrows).

9.2 Data Size of Record Data

Since the PN/PROFIBUS Linking Device is modelling one or more PROFIBUS lines (depending on the number of the PROFIBUS master systems), the associated IO controller should be selected according its ability of sending a huge volume of configuration data.

10 Requirements for certification tests

The requirements of the PROFINET IO certification are covered by the certification test for PROFINET IO-devices. There are no additional requirements to be considered. The certification of the fieldbus part is covered by the established certification procedure of PROFIBUS International.

PROFIBUS Nutzerorganisation e.V.
Haid-und-Neu-Str. 7
76131 Karlsruhe
Germany
Phone: +49 (0) 721 / 96 58 590
Fax: +49 (0) 721 / 96 58 589
e-mail: info@profibus.com
http://www.profibus.com