Interface Specification
PROFIBUS Ethernet Gateway
FG-100 / FG-300

Version 5.4
Rev. 03

Date: 16-October-2008

Softing AG
Richard-Reitzner-Allee 6
D-85540 Haar
Phone (++49) 89 45 65 6 - 0
Fax (++49) 89 45 65 6 - 399

INHALT

1 OVERVIEW

The “PROFIBUS Client Socket Application Program Interface (API)” is an interface between a user application running on a remote pc and the PROFIBUS protocol software running on a PROFIBUS Ethernet gateway “FG-x00-PB”.
The communication between the remote PC and the FG-x00-PB makes use of TCP/IP. The PROFIBUS Client Socket API uses the standardized socket interface in order to access TCP/IP services.
Section 2 describes the Softing PROFIBUS Application Program Interface (PAPI) with its services and data.
Section 3 outlines the particular functions of the PAPI.
Section 4 shows the communication between the PROFIBUS Client Socket API and the FG-x00-PB in detail.
Section 5 describes the coding of the telegrams of the PAPI.

2 PROFIBUS APPLICATION PROGRAM INTERFACE

The application interface of the Softing - PROFIBUS Application Program Interface (PAPI) – offers two mechanisms for the data exchange between the application and the protocol software.:
- a service interface for service-oriented data exchange.
- a data interface for fast cyclic data exchange

2.1 Service interface

Over the service interface service-oriented data are sent and received. The interface can handle one service between remote PC and FG-x00-PB at a time. Every service is identified unambiguously by the service-independent so-called Service Description Block and the service-dependent data block.

The Service Description Block contains the following parameters:

- `comm_ref` communication reference (logischer Kanal)
- `layer` this parameter indicates the instance that the service addresses.
- `service` identification of the service. Each service has a unique, layer-dependent ID.
- `primitive` type of service
  There are the following service types:
  - `REQ` request (service request of the local user)
  - `CON` confirmation (confirmation of the remote server)
  - `IND` indiction (service request of a remote client)
  - `RES` response (response of the local user)
- `invoke_id` identification of a individual service instance (only for concurrent services)
- `result` `POS / NEG` result of the service, error details are service-dependent and part of the service response

Each service is associated with a service-specific data block. Service-specific data blocks are covered in detail by the user manual “PROFIBUS Application Program Interface”. The following functions are used to send and receive services:

- `profi_snd_req_res`
- `profi_rcv_con_ind`

2.2 Data interface

The data interface serves the fast, cyclic data exchange.

DP master: The output data of a DP slave are set and the input data are read.

DP slave: The input data of the DP slave are set and the output data are read. Furthermore, the current input data can be read back.

The following functions are available for that purpose:

- `profi_set_data`
- `profi_get_data`
- `profi_set_dps_input_data`
- `profi_get_dps_input_data`
- `profi_get_dps_output_data`

With these functions current input and output data are transferred from/to the FG-x00-PB via TCP/IP.

3 PAPI FUNCTIONS

3.1 Initialization

The function **`profi_init()`**initializes the PAPI, establishes a TCP/IP connection to the FG-x00-PB and initializes the/a PROFIBUS unit of the FG-x00-PB.

```
#include "PB_IF.H"
```

Function prototype:

```
INT16 profi_init
(
  IN USIGN8  Board
  IN USIGN32 ReadTimeout,
  IN USIGN32 WriteTimeout
)
```

Parameter:

`Board`	board number
`ReadTimeout`	timeout value for read access
`WriteTimeout`	tmeout value for write access

Return values:	(defined in file PB_ERR.H)
`E_OK`	initialization successful
`E_IF_NO_CNTRL_RES`	no response from the PROFIBUS unit in the FG-x00-PB (Timeout)
`E_IF_INIT_INVALID_PARAMETER`	one or several parametersof the initialization are invalid
`E_IF_INIT_FAILED`	initialization of the PROFIBUS unit in the FG-x00-PB failed
`E_IF_SOCKET_ERROR`	an operating sysem error occurred (the actual error code can be read out with the function `WSAGetLastError()` under Windows or using `errno` under Linux)

3.2 Shutdown

The function profi_end() disconnects the TCP/IP connection to the FG-x00-PB and terminates the PAPI. The PROFIBUS unit in the FG-x00-PB is reset.

```
#include "PB_IF.H"
```

Function prototype:

```
INT16 profi_end
(
  VOID
)
```

Parameters: none

Return values: (defined in file PB_ERR.H)

E_OK	function executed successfully
E_IF_SOCKET_ERROR	an operating sysem error occurred (the actual error code can be read out with the function WSAGetLastError() under Windows or using errno under Linux)

3.3 Connection monitoring

The function **profi_snd_idle()** is used to monitor the TCP/IP connection. It must be called periodically by the application. If the FG-x00-PB does not receive an idle telegram within a specified period, the TCP/IP connection will be closed automatically. The PROFIBUS unit in the FG-x00-PB is reset.

```
#include "PB_IF.H"
```

Function prototype:

```
INT16 profi_snd_idle
(
  VOID
)
```

Parameters: none

Return values: (defined in file PB_ERR.H)

E_OK	function executed successfully
E_IF_SOCKET_ERROR	an operating sysem error occurred (the actual error code can be read out with the function WSAGetLastError() under Windows or using errno under Linux)

3.4 Send service request or service response

The function **`profi_snd_req_res()`** transfers request or response data to the FG-xxxPB via TCP/IP and mandates the PROFIBUS unit of the FG-x00-PB to send a service request or a service response to the PROFIBUS network.

```
#include "PB_IF.H"
```

Function prototype:

```
INT16 profi_snd_req_res
(
  IN  T_PROFI_SERVICE_DESCR*  pSdb,
  IN  VOID*                   pData,
  IN  PB_BOOL                 Dummy
)
```

Parameters:

`pSdb`	pointer to the Service Description Block (data structure T_PROFI_SERVICE_DESCR)
`pData`	pointer to a variable with the service-specific data structure
`Dummy`	reserved for internal use

Return values:	(defined in file PB_ERR.H)
`E_OK`	service request or service response issued successfully
`E_IF_INVALID_LAYER`	invalid service layer
`E_IF_INVALID_SERVICE`	invalid serivce
`E_IF_INVALID_PRIMITIVE`	invalid serivce type
`E_IF_RESOURCE_UNAVAILABLE`	no resources available to transfer the service
`E_IF_NO_PARALLEL_SERVICES`	concurrent services of the same type are not allowed
`E_IF_SERVICE_CONSTR_CONFLICT`	service is currently not available, retry later
`E_IF_SERVICE_NOT_SUPPORTED`	this service is not supported
`E_IF_SERVICE_NOT_EXECUTABLE`	service can not be executed (reason unknown, mostly: wrong state).
`E_IF_NO_CNTRL_RES`	the PROFIBUS unit does not respond (timeout).
`E_IF_INVALID_DATA_SIZE`	not enough memory to process the request / the response or wrong length entry
`E_IF_INVALID_PARAMETER`	invalid parameter in the request / the resonse
`E_IF_PAPI_NOT_INITIALIZED`	PAPI is not initialized, the function profi_init() has not been called

E_IF_SOCKET_ERROR	an operating sysem error occurred (the actual error code can be read out with the function WSAGetLastError()under Windows or using errno under Linux)

3.5 Receive service confirmation or service indication

The function **`profi_rcv_con_ind()`** transfers a service confirmation or a service indication via TCP/IP that has been received by the PROFIBUS unit of the FG-x00-PB.

```
#include "PB_IF.H"
```

Function prototype:

```
INT16 profi_rcv_con_ind
(
  INOUT T_PROFI_SERVICE_DESCR*  pSdb,
  INOUT VOID*                   pData,
  INOUT USIGN16                 pDataLength
)
```

Parameters:

`pSdb`	(IN) pointer to the Service Description Block (data structure `T_PROFI_SERVICE_DESCR`) (OUT) completed Service Description Block
`pData`	(IN) pointer to the data buffer (OUT) completed data block of the service
`pDataLength`	(IN) pointer to an USIGN16 variable containing the length of the data buffer (OUT) length of the received data

Return values:	(defined in file PB_ERR.H)
`CON_IND_RECEIVED`	a confirmation or an indication of a service has been received
`NO_CON_IND_RECEIVED`	neither a confirmation nor an indication of a service has been received
`E_IF_NO_CNTRL_RES`	the PROFIBUS unit does not respond (timeout).
`E_IF_FATAL_ERROR`	data buffer has been filled with `T_EXCEPTION`
`E_IF_INVALID_DATA_SIZE`	buffer too small
`E_IF_PAPI_NOT_INITIALIZED`	PAPI is not initialized, the function profi_init() has not been called
`E_IF_SOCKET_ERROR`	an operating sysem error occurred (the actual error code can be read out with the function `WSAGetLastError()`under Windows or using `errno` under Linux)

3.6 Writing output data

The function **`profi_set_data()`** sends output data via TCP/IP to the FG-x00-PB. These data are written into the dual port RAM of its PROFIBUS unit.

```
INT16 profi_set_data
(
  IN USIGN8      DataId,
  IN USIGN16     Offset,
  IN USIGN16     DataSize,
  IN VOID        pData
);
```

Parameters:

`DataId:`	identifier of the slave i/o data image (DP)
`Offset:`	offset in the slave i/o data image
`DataSize:`	number of output data to be written
`pData:`	pointer to the output data to be written

Possible values of the parameter DataId (defined in file PB_IF.H)

`ID_DP_SLAVE_IO_IMAGE`	identifier of the slave i/o data image (DP)

A detailed description of the slave i/o data image can be found in the manual “PROFIBUS Application Program Interface” in section “DP services”.

Return values:	(defined in file PB_ERR.H)
`E_OK`	output data successfully set
`E_IF_SERVICE_NOT_SUPPORTED`	service not supported, i/o data image not found
`E_IF_INVALID_DATA_SIZE`	the number of data to be written does not fit the configured number of output data
`E_IF_NO_CNTRL_RES`	the PROFIBUS unit does not respond (timeout).
`E_IF_PAPI_NOT_INITIALIZED`	PAPI is not initialized, the function profi_init() has not been called
`E_IF_INVALID_DP_STATE`	the DP master is not in state OPERATE, output data cannot be written
`E_IF_SOCKET_ERROR`	an operating sysem error occurred (the actual error code can be read out with the function `WSAGetLastError()` under Windows or using `errno` under Linux)

3.7 Reading input data

The function **`profi_get_data()`** reads input data via TCP/IP from ther PROFIBUS unit of the FG-x00-PB.

```
INT16 profi_get_data
(
  IN    USIGN8    DataId,
  IN    USIGN16   Offset,
  INOUT USIGN16*  pDataSize,
  OUT   VOID      pData,
);
```

Parameters:

`DataId:`	identifier of the slave i/o data image (DP)
`Offset:`	offset in the slave i/o data image
`pDataSize:`	(IN) pointer to an USIGN16 variable containing the length of the data buffer (OUT) length of the input data to be read
`pData:`	pointer to the input data to be read

Possible values of the parameter DataId (defined in file PB_IF.H)

`ID_DP_SLAVE_IO_IMAGE`	identifier of the slave i/o data image (DP)

A detailed description of the slave i/o data image can be found in the manual “PROFIBUS Application Program Interface” in section “DP services”.

Return values:	(defined in file PB_ERR.H)
`E_OK`	input data successfully read
`E_IF_SERVICE_NOT_SUPPORTED`	service not supported, i/o data image not found
`E_IF_INVALID_DATA_SIZE`	buffer too small
`E_IF_NO_CNTRL_RES`	the PROFIBUS unit does not respond (timeout).
`E_IF_PAPI_NOT_INITIALIZED`	PAPI is not initialized, the function profi_init() has not been called
`E_IF_INVALID_DP_STATE`	the DP master is not in state OPERATE, input data cannot be read
`E_IF_SOCKET_ERROR`	an operating sysem error occurred (the actual error code can be read out with the function `WSAGetLastError()` under Windows or using `errno` under Linux)

3.8 Setting input values for slave operation

The function **`profi_set_dps_input_data()`** sends input data for the DP slave via TCP/IP to the FG-x00-PB. These data are written into the dual port RAM of its PROFIBUS unit. Always an entire set of slave input data is written.

```
#include "PB_IF.H"
```

Function prototype:

```
INT16 profi_set_dps_input_data
(
  IN  USIGN8*  pData,
  IN  USIGN8   pDataSize,
  OUT USIGN8*  pState
)
```

Parameter:

`pData:` pointer to a USIGN8 variable that holds the input data

`pDataSize:` number of input data (in bytes) to be written, if this number does not match the configured length of input data, the error '`E_IF_INVALID_DATA_SIZE`' is returned

`pState:` pointer to the current input data status bit field with:

`DPS_INPUT_STATE_FREEZE_ENABLED`
: the slave has activated the function to freeze its inputs

`DPS_INPUT_STATE_FREEZE_COMMAND`
: a corresponding Global_Control command was received. The input data supplied with the latest call of function '`profi_set_dps_input_data`' is stored as current transmit data of the slave. After completion of this function the bit is reset automatically.

Return values: (defined in file PB_ERR.H)

`E_OK`	input data successfully set and the status reported
`E_IF_SERVICE_NOT_SUPPORTED`	service not supported, i/o data image not found
`E_IF_INVALID_DATA_SIZE`	the number of data to be written does not fit the configured number of input data
`E_IF_NO_CNTRL_RES`	the PROFIBUS unit does not respond (timeout).
`E_IF_PAPI_NOT_INITIALIZED`	PAPI is not initialized, the function profi_init() has not been called
`E_IF_SOCKET_ERROR`	an operating sysem error occurred (the actual error code can be read out with the function `WSAGetLastError()` under Windows or using `errno` under Linux)

3.9 Reading back the input values of the slave

The function **`profi_get_dps_input_data()`** reads back the current inputs of the slave and its corresponding status via TCP/IP from the PROFIBUS unit of the FG-x00-PB.

```
#include "PB_IF.H"
```

Function prototype:

```
INT16 profi_get_dps_input_data
(
  OUT   USIGN8*  pData,
  INOUT USIGN8*  pDataSize,
  OUT   USIGN8*  pState
)
```

Parameters:

`pData` (OUT) input data of the slave

`pDataSize:` (IN) pointer to an USIGN16 variable containing the length of the data buffer
(OUT) length of the input data read

`pState:` pointer to the current input data status bit field with:

`DPS_INPUT_STATE_FREEZE_ENABLED`
the slave has activated the function to freeze its inputs

`DPS_INPUT_STATE_FREEZE_COMMAND`
a corresponding Global_Control command was received. The input data supplied with the latest call of function '`profi_set_dps_input_data`' is stored as current transmit data of the slave. The status is read only. The bit is reset only after the next call of function '`profi_set_dps_input_data`'.

Return values:	(defined in file PB_ERR.H)
`E_OK`	input data successfully set and the status reported
`E_IF_SERVICE_NOT_SUPPORTED`	service not supported, i/o data image not found
`E_IF_INVALID_DATA_SIZE`	buffer too small
`E_IF_NO_CNTRL_RES`	the PROFIBUS unit does not respond (timeout).
`E_IF_PAPI_NOT_INITIALIZED`	PAPI is not initialized, the function profi_init() has not been called
`E_IF_SOCKET_ERROR`	an operating sysem error occurred (the actual error code can be read out with the function `WSAGetLastError()`under Windows or using `errno` under Linux)

3.10 Reading the output values of the slave

The function **`profi_get_dps_output_data()`** reads the current output values of the DP slave via TCP/IP from the dual port RAM of the PROFIBUS unit of the FG-x00-PB .

```
#include "PB_IF.H"
```

Function prototype:

```
INT16 profi_get_dps_output_data
(
  OUT   USIGN8* pData,
  INOUT USIGN8* pDataSize,
  OUT   USIGN8* pState
)
```

Parameters:

`pData` pointer to a USIGN8 variable array to read the outputs of the slave

`pDataLength` (IN) pointer to a USIGN8 variable indicating the buffer size in bytes
(OUT) number of output data read

`pState` pointer to the current output data status bit field with:

`DPS_OUTPUT_STATE_SYNC_ENABLED`
the function for freezing the outputs has been enabled.

`DPS_OUTPUT_STATE_SYNC_COMMAND`
a corresponding Global_Control command was received. Since the last time the function ‘`profi_get_dps_output_data`‘ was called, a Sync command has been received upon which received upon which new output data have been made ready. The bit is cleared automatically after access.

`DPS_OUTPUT_STATE_CLEAR_DATA`
The outputs are in failsafe state. A corresponding command was received from the master.

`DPS_OUTPUT_STATE_VALID_DATA`
No transmission errors have occurred during data transmission from the master and user data are exchanged (no timeout or watchdog error).

`DPS_OUTPUT_STATE_NEW_DATA`
New output data were received from the master. Since the last access via ‘`profi_get_dps_output_data`‘ function new data have been delivered (independent of the Sync command). With this bit you can prevent reusing old data. The bit is cleared after access.

`DPS_OUTPUT_STATE_GLOBAL_CONTROL`
Since the last time the output data were read, a Global_Control command has been received. The bit is cleared as soon as the output data have been read.

Return values: (defined in file PB_ERR.H)

E_OK	output data successfully read and the status reported
E_IF_SERVICE_CONSTR_CONFLICT	service can not be executed in current context, synchronization error with PROFIBUS unit
E_IF_SERVICE_NOT_SUPPORTED	service not supported, i/o data image not found
E_IF_INVALID_DATA_SIZE	buffer too small
E_IF_NO_CNTRL_RES	the PROFIBUS unit does not respond (timeout).
E_IF_PAPI_NOT_INITIALIZED	PAPI is not initialized, the function profi_init() has not been called
E_IF_SOCKET_ERROR	an operating sysem error occurred (the actual error code can be read out with the function WSAGetLastError()under Windows or using errno under Linux)

3.11 Version information

The function **`profi_get_versions()`** retrieves information about the firmware version and PAPI version from the FG-x00-PB.

```
#include "PB_IF.H"
```

Function prototype:

```
INT16 profi_get_versions
(
  INOUT CSTRING* pPapiVersion,
  INOUT CSTRING* pFirmwareVersion
)
```

Parameter:

`pPapiVersion`	(IN) (OUT)	pointer to CSTRING variable containing the current PAPI version string containing the current PAPI version
`pFirmwareVersion`	(IN) (OUT)	pointer to CSTRING variable containing the current firmware version string containing the current firmware version

Caution: for each pointer a buffer must be reserved with a minimum length of `VERSION_STRING_LENGTH`

Return values: (defined in file PB_ERR.H)

`E_OK`	version of PAPI anfd version of firmware successfully read
`E_IF_NO_CNTRL_RES`	the PROFIBUS unit does not respond (timeout).
`E_IF_PAPI_NOT_INITIALIZED`	PAPI is not initialized, the function profi_init() has not been called
`E_IF_SOCKET_ERROR`	an operating sysem error occurred (the actual error code can be read out with the function `WSAGetLastError()`under Windows or using `errno` under Linux)

3.12 Serial number information

The function **`profi_get_serial_device_number()`** retrieves the serial number of the PROFIBUS unit in the FG-x00-PB.

```
#include "PB_IF.H"
```

Function prototype:

```
INT16 profi_get_serial_device_number
(
  OUT USIGN32* pSerialDeviceNumber
)
```

Parameter:

`pSerialDeviceNumber`	serial number of the PROFIBUS unit

Return values:	(defined in file PB_ERR.H)
`E_OK`	serial number successfully read
`E_IF_NO_CNTRL_RES`	the PROFIBUS unit does not respond (timeout).
`E_IF_PAPI_NOT_INITIALIZED`	PAPI is not initialized, the function profi_init() has not been called
`E_IF_SOCKET_ERROR`	an operating sysem error occurred (the actual error code can be read out with the function `WSAGetLastError()` under Windows or using `errno` under Linux)

4 COMMUNICATION BETWEEN REMOTE PC AND FG-X00-PB

The communication between a remote PC and the FG-x00-PB makes use of the TCP/IP protocol. The so-called “socket interface” serves as standardized programming interface between the application program and TCP/IP.

4.1 Socket interface

Using “sockets” facilitates the establishment and utilization of a TCP/IP connection by an application program. The application software can use a set of system calls for TCP/IP communication. The following socket functions are necessary for the communication between a raomote client PC and the FG-x00-PB.

- socket()	create socket
- connect()	establish connection
- send()	send data
- recv()	receive data
- close()	close socket

In order to establish a connection to the FG-x00-PB the application software call the function `socket()`to create an instance of the socket interface and to define the transport protocol. The function `connect()` allows the application software to connect to the FG-x00-PB. Now, the remote PC and the FG-x00-PB can execute byte-oriented data exchange with the function `send()` and `recv()`.Finally, the function `close()` terminates the connection.
Detailled information about the socket interface and its utilization can be found in relevant technical literature.

4.2 Communication between PAPI client and FG-x00-PB

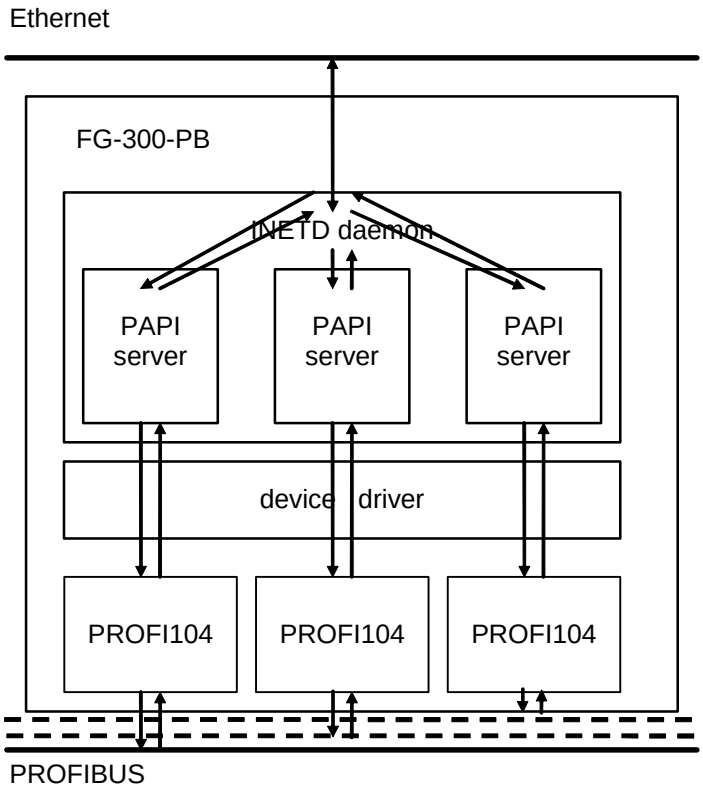
The PAPI-Client and the FG-x00-PB act in a client-server relationship. The client sends a request telegram to the server. The server processes this request telegram and confirms it with a response telegram containing a return value and the requested data.

The server in the FG-x00-PB is structured hierarchically. Connections are established via the PAPI-PORT of the server. For each connection requested by the client the server starts a respective child process that from then on takes over the processing of telegrams received over this connection. During initialization each child process requires information about the actual PROFIBUS unit it is assigned to. From then on communication with a particular PROFIBUS unit is only possible over the respective connection.

The hardware and software configuration of the FG-x00-PB is supplied by its integrated HTTP server. This HTTP server can be queried at any time.

4.3 Connection establishment and FG-x00-PB hierarchy

A remote PAPI client connects to the FG-x00-PB over the PAPI-PORT (**2356**). The FG-x00-PB's internal INETD daemon starts the respective PAPI server and hands over the control of the connection to this server while closing its own connection to the PAPI client. Afterwards the INETD daemon is ready to accept a new connection request from thenext PAPI client.



4.4 Communication between PAPI client and HTTP server

The FG-x00-PB's configuration can be requested by the PAPI client by sending the message '**GET /bin/config HTTP/1.0\n\n'** to the FG-x00-PB's HTTP server.

The response contains the unambiguous XML configuration string of the respective FG-x00-PB.

```
<config>
   <serial number>0021200127</serial number>
   <firmware version>"fg300pb_V1.02.0.09.release Tue Dec 5 11:56:35 CET 2006"</firmware version>
   <network>
         <mac address>00:06:71:06:00:7F</mac address>
         <network name>FG_100_PB</network name>
         <network address>172.17.10.177</network address>
         <maintenance address>192.168.212.231</maintenancek address>
         <network gateway>172.17.0.160</network gateway>
         <device port>2357</device port>
         <papi port>2356</papi port>
   </network>
   <fieldbus>
         <number of boards> 1 </number of boards>
         <driver version>PROFIboard Driver V1.03.0.01.release </driver version >
         <node>
               <number> 0 </number>
               <type> PROFI104 </type>
               <serial number> 000901497 </serial number>
               <firmware version> PROFI104 DP/FMS 5.27.0.00.release</firmware version>
               <board device>
                     <name> /dev/pbboard0 </name>
               </board device>
               <service device>
                     <name> /dev/pbservice0 </name>
               </service device>
               <data device>
                     <name> /dev/pbdata0 </name>
               </data device>
               <slave input device>
                     <name> /dev/pbslin0 </name>
               </slave input device>
               <slave output device>
                     <name> /dev/pbslout0 </name>
               </slave output device>
         </node>
   </fieldbus>
</config>
```

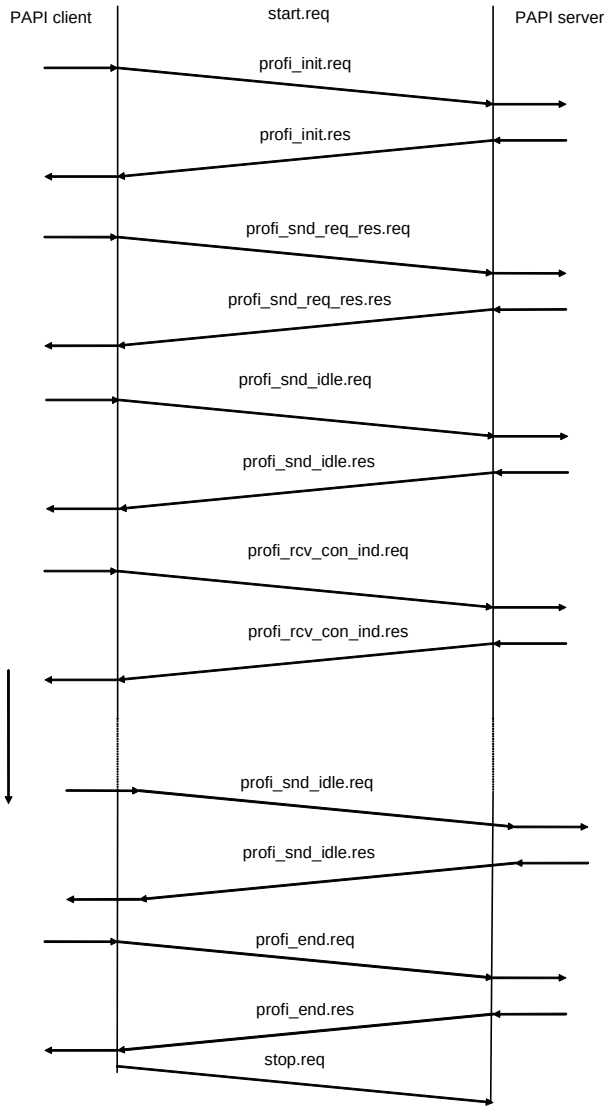
The section <node>...</node> is repeated for every PROFIBUS unit present in the FG-x00-PB. All data are ASCII coded.

4.5 Communication between PAPI client and PAPI server

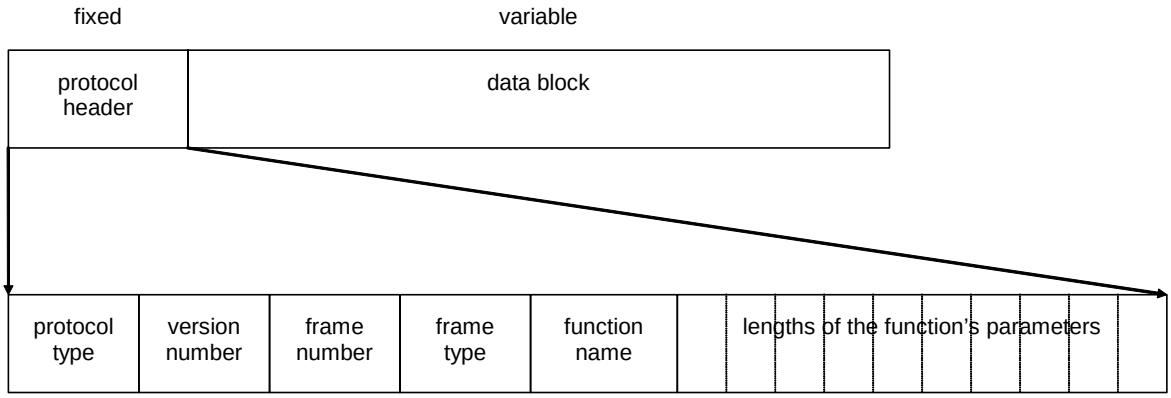
The client sends requests to the server. The server processes this request telegram and confirms it with a response telegram containing a return value and the requested data.

The health of the Ethernet connection is monitored. For that purpose the PAPI client periodically sends `profi_snd_idle.req` telegrams.

The server recognizes each packet received from the client and retriggers its local watchdog accordingly. The timeout value is supplied with the function `profi_init.` It is greater or equal double the time after which the client needs to send a `profi_snd_idle.req` telegram. If no telegram is received within this period the server is terminated.



5 Coding of the telegrams



The protocol header is coded as follows:

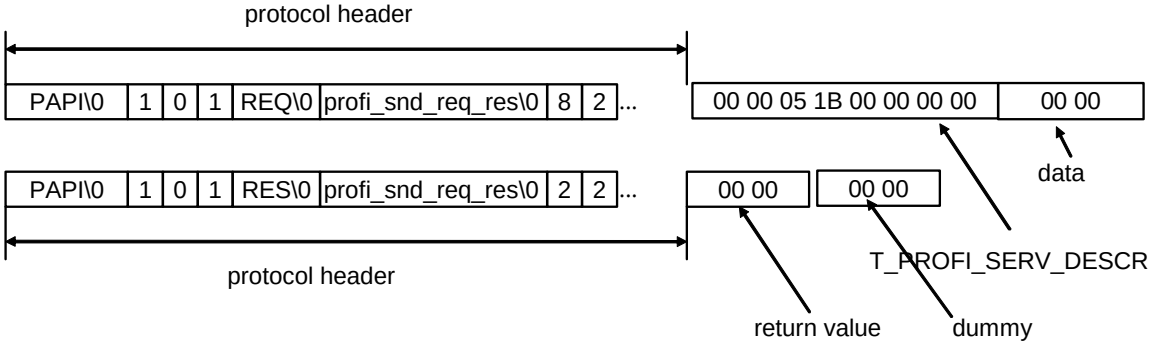
- protocol type: PAPI (string of fixed length 10 bytes, string is terminated by '\0')
- version number of the remote client / server (2 bytes, major number changes if functionality changes, minor number indicates version changes with maintained compatibility)
- frame number contains the number of the frame to be transferred (2 bytes)
- frame type: request / response (4 bytes, 'REQ\0', 'RES\0' or 'IND\0')
- function name (string of fixed length 40 bytes, string is terminated by '\0')
- lengths of the function's parameters (fixed count: 10 parameters, each function knows the actual number of parameters used)

Protocol type, frame type and function name are strings. Version number, frame number and length information use binary format.

The data block contains the function's parameters in binary format. The data are not separated by any special characters. They are simply attached to each other.

5.1 Sample telegram

PAPI: service "send request"



5.2 Coding of PAPI telegrams

function	primi-tive	number of parameters	parameter description type	parameter
profi_init	REQ	4	- USIGN8	BusConnector
			- USIGN32	ReadTimeout
			- USIGN32	WriteTimeout
			- USIGN32	WatchdogTimeout
profi_init	RES	2	- INT16	ReturnValue
			- INT16	DummyValue
profi_end	REQ	0		
profi_end	RES	2	- INT16	ReturnValue
			- INT16	DummyValue
profi_snd_idle	REQ	0		
profi_snd_idle	RES	2	- INT16	ReturnValue
			- INT16	DummyValue
profi_snd_req_res	REQ	2	- T_SDB	ServiceDescriptionBlock
			- BYTE-ARRAY	Data
profi_snd_req_res	RES	2	- INT16	ReturnValue
			- INT16	DummyValue
profi_rcv_con_ind	REQ	1	- USIGN16	BufferSize
profi_rcv_con_ind	RES	5	- INT16	ReturnValue
			- INT16	DummyValue
			- T_SDB	ServiceDescriptionBlock
			- USIGN16	DataSize
			- BYTE-ARRAY	Data
profi_set_data	REQ	4	- USIGN8	DataId
			- USIGN16	Offset
			- USIGN16	DataSize
			- BYTE-ARRAY	Data
profi_set_data	RES	2	- INT16	ReturnValue
			- INT16	DummyValue
profi_get_data	REQ	3	- USIGN8	DataId
			- USIGN16	Offset
			- USIGN16	BufferSize
profi_get_data	RES	4	- INT16	ReturnValue
			- INT16	DummyValue
			- USIGN16	DataSize
			- BYTE-ARRAY	Data
profi_set_dps_input_data	REQ	2	- USIGN8	DataSize
			- BYTE-ARRAY	Data
profi_set_dps_input_data	RES	3	- INT16	ReturnValue
			- INT16	DummyValue
			- USIGN8	State

profi_get_dps_input_data	REQ	1	- USIGN8	BufferSize
profi_get_dps_input_data	RES	5	- INT16 - INT16 - USIGN8 - BYTE-ARRAY - USIGN8	ReturnValue DummyValue DataSize Data State
profi_get_dps_output_data	REQ	1	- USIGN8	BufferSize
profi_get_dps_output_data	RES	5	- INT16 - INT16 - USIGN8 - BYTE-ARRAY - USIGN8	ReturnValue DummyValue DataSize Data State
profi_get_versions	REQ	0		
profi_get_versions	RES	4	- INT16 - INT16 - CSTRING - CSTRING	ReturnValue DummyValue PapiVersion FirmwareVersion
profi_get_serial_device_number	REQ	0		
profi_get_serial_device_number	RES	3	- INT16 - INT16 - USIGN32	ReturnValue DummyValue SerialDeviceNumber
profi_get_last_error	REQ	0		
profi_get_last_error	RES	3	- INT16 - INT16 - INT16	ReturnValue DummyValue LastError