
TURCK

Industrial Automation

# USER MANUAL RFID SYSTEM

# SET-UP IN PROFINET WITH THE PROXY IDENT FUNCTION BLOCK





BL ident®

Edition 03/08

# 0 About this manual



# 1 The TURCK *BL ident*® system



# 2 Commissioning a TURCK *BL ident*® system

## 3 Extract from the specification



## 4 Glossary

# Safety Instructions!

### Before beginning installation work

- Disconnect the device from the power supply
- Ensure against accidental restart
- Verify isolation from the supply
- Earth and short-circuit the supply
- Cover or close off neighbouring units that are live.
- The assembly instructions provided for the device are to be complied with.
- Only suitably qualified personnel according to EN 50 110-1/-2 (VDE 0105 part 100) are authorised to carry out work on this device/system.
- When conducting installation work ensure that you are free of electrostatic charge before touching the device.
- The functional earth (FE) must be connected to the protective earth (PE) or the equipotential bonding. The system installer is responsible for establishing this connection.
- Connection and signal cables are to be installed so that any inductive or capacitive interference does not impair the automation functions.
- The installation of automation devices and their operating elements is to be carried out in such a way as to prevent unintentional operation.
- In order to prevent cable or wire breakage on the signal side generating undefined states in the automation devices, appropriate safety measures are to be taken for the I/O coupling on the hardware and software side.
- Ensure a reliable isolation of the extra-low voltage for the 24 volt supply. Only those power supply units that comply with IEC 60 364-4-41, i.e. HD 384.4.41 S2 (VDE 0100 part 410) are to be deployed.
- Fluctuations or deviations of the mains voltage from the nominal value should not exceed the tolerance limits specified in the technical data, otherwise malfunctions and dangerous states may occur.
- Emergency stop devices complying with IEC/EN 60 204-1 must remain effective in all operating modes of the automation installation. Releasing the emergency stop devices must not cause a restart.
- Devices for mounting in housings or cabinets, desktop or portable units, are only to be operated and controlled with the housing closed.
- Measures are to be taken to ensure the correct restarting of a program following interruption due to a voltage drop or failure. Dangerous operating conditions, even short term, should not occur as a result. If required an emergency stop should be carried out.
- External measures are to be implemented at those locations where faults in the automation installation could lead to injury to persons or damage to property. These measures must guarantee safe operating conditions even in the event of a fault or malfunction (e.g. by means of independent limit switches or mechanical locking devices etc.).
- The electrical installation must be carried out in accordance with the relevant regulations (e.g. in respect of the cable cross sections, uses and protective earth connections).
- All work involving transport, installation, commissioning and maintenance is to be carried out exclusively by qualified personnel. (in accordance with IEC 60 364 i.e. HD 384 or DIN VDE 0100 and national accident prevention regulations).
- All covers and doors must be kept closed during operation..

# 0 About this manual

## Documentation concept

In the first chapter of this manual you are provided with an overview of the TURCK *BL ident*® system.

The second chapter contains instructions for commissioning of a *BL ident*® systems using the standard function block “Proxy Ident Function Block”. The commissioning example is undertaken using a SIMATIC S7/-300 station (Siemens). SIMATIC STEP 7 standard software is used.

The third chapter contains an excerpt from the specification “Proxy Ident Function Block”.

## Description of symbols used

**Warning**

This sign can be found next to all notes that indicate a source of hazards. This can refer to danger to personnel or damage to the system (hardware and software). This sign means for the operator: work with extreme caution.

**Attention**

This sign can be found next to all notes that indicate a source of potential hazards. This can refer to possible danger to personnel or damage to the system (hardware and software) and facility.

**Note**

This sign can be found next to all general notes that supply important information about one or more operating stages.

These specific notes are intended to make operation easier and avoid unnecessary work due to incorrect operation.

**Attention**

Please read this section carefully. Safety aspects cannot be left to chance when dealing with electrical equipment

This manual contains all the necessary information concerning the intended usage of TURCK *BL ident* ® system.
It has been specially developed for qualified personnel who have the required level of expertise.

### Prescribed Use

**Warning**

The devices described in this manual must be used only in applications prescribed in this manual or in the respective technical descriptions, and only in connection with components and devices from third party manufacturers that have been certified.

Appropriate transport, storage deployment and mounting as well as careful operating and thorough maintenance guarantee trouble-free and safe operation of these devices.

### Notes concerning planning / Installation of this product

**Warning**

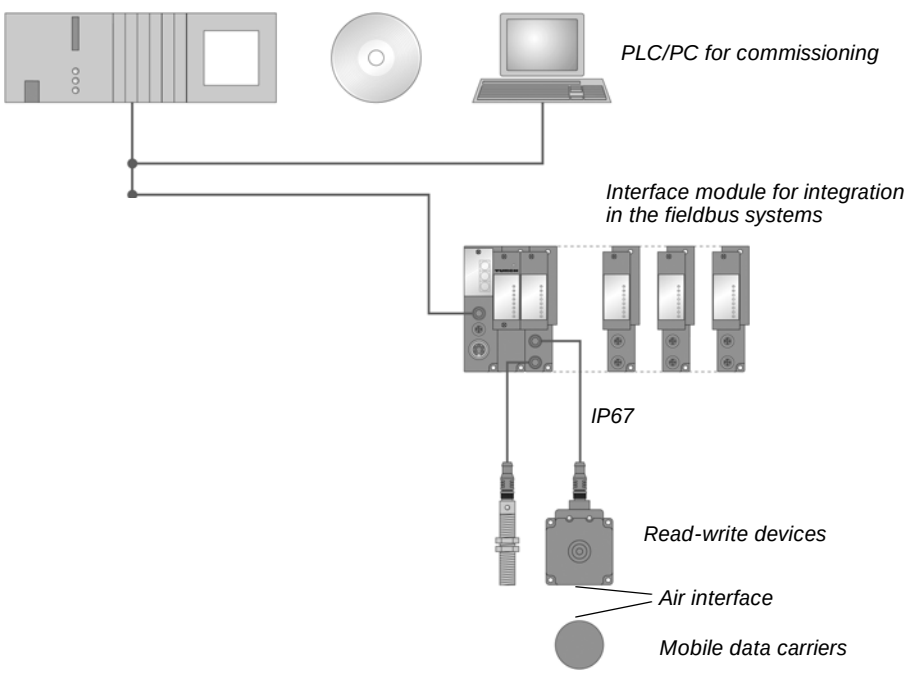
It is imperative that all respective safety measures and accident protection guidelines be adhered to.

# 1 The TURCK *BL ident*® system

## Schematic representation of the identification system *BL ident*®

The TURCK *BL ident*® system consists of several levels. Every level offers variation options. An application adapted to the overall system is possible.

*Figure 1: System overview*




### Support for *BL ident*®- projects

Further support can be found in the following software and documents for engineering, installation and commissioning:

- Available free-of-charge on the Internet for simulation and optimisation of an application at http://www.turck.com.. a "*BL ident*® simulator" is provided.
- D101583 - "Installation of the *BL ident*®-System" - This manual presents the technical details of available TURCK data carriers and the TURCK read-write devices.
- D101581 - "Interface Module for Fieldbus Connection" - This manual describes the professional application of *BL ident*® interface modules.
- D101607 - This manual contains the software description for the so-called "Handheld" (programming device) which can be used to read and write data irrespective of the location.
- D101585 - This manual contains the hardware description for the so-called "Handheld" (programming device) which can be used to read and write data irrespective of the location.
- D101640 - "Commissioning with the software CoDeSys for programmable gateways"
- D101642 - "Set-up with DeviceNet™"
- D101644 - "Set-up with EtherNet/IP"
- D101579 - "Set-up in PROFIBUS-DP with the PROXY IDENT FUNCTION BLOCK"
- D300890 - „User manual for PROFINET IO"

### Networking with *BL ident*® systems

As it is possible to integrate *BL ident*® systems in (existing) bus systems, networking of several *BL ident* systems is possible.

The guidelines which relate to the maximum extension of the respective bus systems apply.

## Identification systems with radio frequency technology (RFID)

RFID is the abbreviation for Radio Frequency Identification.

An RFID system consists of a data carrier, a device for reading and writing the data from the data carrier as well as other devices which perform the transfer and processing of data.

The transfer of data from the data carrier to the read-write heads is undertaken without contact using electromagnetic waves. This type of data transfer is insensitive to dirt and temperature fluctuations.

The data carriers can be attached directly to a product. The term “Mobile data carriers” is used for this reason. Further terms used for the data carriers are TAGs or transponders. The data content can consist of production and manufacturing data. Important it that this data identifies the product. This is the origination of the term “Identification System”.

A whole range of possibilities exist as the data content can be changed by writing on the data carrier. Accordingly, the production / manufacturing processes can be traced and monitored. Logistics/distribution can be optimised.

The “Identification Systems” can be integrated into (existing) fieldbus automation systems (e.g. PROFINET IO). The integration of the respective fieldbus system is undertaken with suitable interface modules.

Standardized software modules (e.g. the Proxy Ident Function Block for PROFINET IO) enable simple system integration and commissioning.

## Characteristics and fields of application of the *BL ident*® system

In order to comply with the demands presented by different fields of application, TURCK offers the *BL ident*® system with a whole range of combination possibilities of data carriers and read-write heads as well as interface modules for integration into automation systems (e.g. PROFINET IO). Software modules enable simple integration and commissioning.

The characteristics of the TURCK *BL ident*®system are listed in the following:

### Degree of protection

Some data carriers as well as the suitable write-read heads feature a high mechanical degree of protection (e.g. **IP67**) and can thus be subject to the most harsh industrial conditions.

The read-write heads are also available in IP69K (wash-down version).

The integration into a fieldbus system is implemented with suitable TURCK interface modules. The interface modules are available in degrees of protection IP67. TURCK connection cables featuring an adequate degree of protection round off the identification system.

Temperature resistant data carries up to 210°C are available for the high temperature range.

Self-adhesive labels made of paper or film have degree of protection IP40.

### Service life

The service life results from the possible number or read-write operations on the data carriers.

FRAM data carriers feature an **unlimited** number or read operations and $10^{10}$ write operations.

EEPROM data carriers feature an **unlimited** number or read operations and $10^5$ write operations.

The data carriers do not require batteries.

### Transmission frequency

The TURCK *BL ident*® system operates with **13.56 MHz** transmission frequency between the data memories and the read/write devices. Systems which operate with these transmission frequencies are practically immune to electromagnetic interference. The 13.56 MHz transmission frequency has developed into a standard in many RFID fields of application.

### Housing styles

### Data carriers

TURCK supplies round, flat data media e.g. with diameters of 16, 20, 30 and 50 mm.

The high temperature data carriers have a cylindrical design (e.g. 22 x 125 mm,).

Inlays and labels have film thickness (size e.g. 43 x 43 mm).

Special housing designs are intended for mounting in and on metal. Some data carriers have holes so that they can be attached with screws.

TURCK can supply customized data carrier solutions on request.

### Read-write heads

The read-write heads are available in different housing styles: Threaded barrels (EM18, EM30, M18 M30), rectangular (CK40, Q80, Q350), ring sensors (S32XL).

### Memory capacity

The memory capacity of the data carrier is **256/128/64 bytes (240/112/48 bytes user data)** with an EEPROM memory device and **2 kilobytes (2000 bytes user data)** with an FRAM memory device.

FRAM: (Ferroelectric Random Access Memory), non-volatile, high service life based on the higher number of write-read operations and faster write operations than EEPROM.

EEPROM: (Electrically erasable programmable read only memory), non-volatile.

Many data carrier variants (EEPROM and FRAM) comply with the ISO-15693 standard.

### Speed of the data carrier relative to the read-write head

**Note**

The speed with which the data carrier can move past the read-write head is influenced by the amount of data to be processed and varies depending on the combination of read-write head and data carrier used.
The figures stated as the maximum speed and data quantities can therefore only serve as examples!

The speed with which the data carrier can move past the read-write head can for example, be increased with the data carrier TW-R50-K2 and the read-write head TN-CK40-H1147 up to **2.5 m/s for 8 bytes** at a distance of 36 mm.

Using the "*BL ident*® simulator" the application variables speed, range and data quantity can be varied. The optimum combination can be selected for the respective combination of read-write head and data carrier. The „*BL ident*® simulator" is available online at http://www.turck.com.. Please observe the restrictive notes in this section.

**Note**

In addition to the data processing times in the read-write heads, the processing time in the overall identification system structure must be considered („System overview" page 1-2). The time for transfer and processing of the data in the overall system can deviate from application to application!
If your application intends using a quick succession of data carriers, it may be necessary to reduce the speed at which the data carriers pass the read-write heads. If in doubt, we recommend determining the speed empirically!

**Note**

The data transfer curves (maximum read/write distances, length of the transfer zone) are only typical values determined under laboratory conditions.
The achievable distances can deviate by up to 30 % due to component tolerances, installation conditions in the application, ambient conditions and influence due to materials (particularly metals).
Therefore a test of the application (particularly when reading and writing in motion) is essential under actual application conditions!
Furthermore, the recommended distances between the data carriers and the read-write heads should be observed to achieve perfect read/write processes irrespective of possible deviations in the range.
The achievable pass speeds (read and write on the fly) and maximum data quantity which can be transferred will change dependent on the actual data transfer curves for the respective application.

### Reading range / writing range

The achievable read-write distance results depending on the respective combination of data carrier and read-write head. The possible read-write distance is influenced by the data quantities to the read or written and the speed at which the data carrier passes the read-write heads. The largest range (**approx. 500 mm**) is achieved using the largest read-write head housing style TNLR-Q350-H1147 and the largest data carrier (Ø 50 mm).
Using the "*BL ident*® simulator" the application variables speed, data quantity and range can be varied. The optimum combination can be selected for the respective combination of read-write head and data carrier. The „*BL ident*® simulator" is available online at http://www.turck.com..

*Figure 2: BL ident® simulator*

### Compatibility

All technical data refers to the *BL ident*® system, i.e. to the combinations of *BL ident*® data carriers, read-write heads and interface modules. Completely different values may apply for data carriers from other manufacturers. Therefore they may only be used after prior approval by TURCK.

### Areas of application (examples)

The characteristics as stated beforehand allow the application of a TURCK *BL ident*® system in the following fields:

- Automotive
- Transport and handling
- Machine (mechanical) engineering
- Food and beverages
- Chemicals
- Pharmaceuticals and petrochemicals.

This includes possible application in all areas such as:

- Assembly lines
- Conveyors
- Industrial manufacturing
- Warehousing
- Logistics
- Distribution
- Component picking
- Transport logistics

# 2 Commissioning a TURCK *BL ident*® system

## Acyclic commissioning example using STEP 7 and PIB

In the following the commissioning of a *BL ident*® using the SIMATIC basis software Step 7 and the standard software block “Proxy Ident Function Block” (PIB) is described.

An initial commissioning should be simple and possible without programming knowledge. TURCK provides an example project for this purpose. You can order the CD “BL IDENT-CD” with the example project directly from TURCK: Ident No. 1545052

### Hardware description of the example project

For the following commissioning example the following hardware components were employed:

- S7 control “CPU 315-2PN/DP” (PROFINET-compatible CPU)
- Gateway “BL67-GW-EN-PN“
- *BL ident*® module BL67-2RFID-A
- *BL ident*® read-write head “TN-CK40-H1147”
- Data carrier “TW-R50-B128” (user data = 112 bytes)

If there are any questions or uncertainties concerning the correct connection or wiring of your *BL ident*®- system, the manuals D101581 (interface modules) and D101583 (read-write heads and data carriers) are available in the download area of the TURCK internet.

### Memory requirements for *BL ident*® commissioning

#### Basic memory requirements

The basic memory requirements for the commissioning of the *BL ident*® system with the Proxy Ident Block are:
**14 kilobytes**

#### Memory requirement per PIB instance (channel)

For each channel an instance of the Proxy Ident Block is generated.

Each channel requires, in addition to the basic memory requirement,
**0,6 kilobytes**.

#### Memory requirements for the read and write data

The Proxy Ident Function Block (PIB) occupies a memory area as send and receive buffer. The size of this memory area must be set in accordance with the data volume generated when reading and writing.

The *BL ident*® system provides TURCK PIB variants in order to cater for the different data volumes generated when reading and writing:

- PIB_1KB
- PIB_16K
- PIB_32K

The following calculation rule shows how you can calculate the memory requirements for reading and writing. It is assumed that the buffer is used by a number of channels/instances. If each instance was to be allocated its own buffer the memory requirements would be considerably higher. Following the completion of the calculation you can select the function block suitable for your application.

The total data volume results from the following sum:

- The volume of data that is read via all active channels and saved to a "read only memory area".
  Once ascertained that the read out of each channels is implemented with sufficient skew, the memory can be divided between the channels.
- The volume of data that is written via all active channels and saved to a "write only memory area".
  Once ascertained that the writing on each channel is implemented with sufficient skew, the memory can be divided between the channels.
- The data volumes that can re-transmit both read and write data. The memory area is used alternatively as write memory area and read memory area. The required memory area can be reduced by half.

**Note**

Please ensure, that transmission time is sufficient in case the read/write memory is used for more channels and/or for reading and writing!

### Memory requirements for the hardware example

The „Hardware description of the example project” page 2-2 provides 2 channels which can each be used for the connection of a read/write unit. The read memory area and the write memory area should be arranged separately.

The data carrier employed can store a maximum of 128 bytes (112 bytes of user data). For the calculation a maximum of 200 bytes of read memory and 200 bytes of write memory have considered. Each channel occupies 400 bytes. Both channels **800 bytes**.

For the **total memory requirement** the „Basic memory requirements” page 2-2 and two times the „Memory requirement per PIB instance (channel)” page 2-2 are added to the 800 bytes:

**Total memory requirement**
**= 14 kilobytes + (2 x 0.6 kilobytes) + 800 bytes = 16 kilobytes**

### Loading the example project and download of the current GSDML file

The example project provided by TURCK enables you to easily reconstruct a first commissioning operation. The example project is available from TURCK on the CD-ROM “BL IDENT-CD” with the Ident-No.1545052

The example project is available as ZIP file. Keep the zipped form and note its location.

The current GSDML file requires it to perform the configuration of the *BL ident* interface module. The current GSDML file can be downloaded at:

http://www.turck.com....

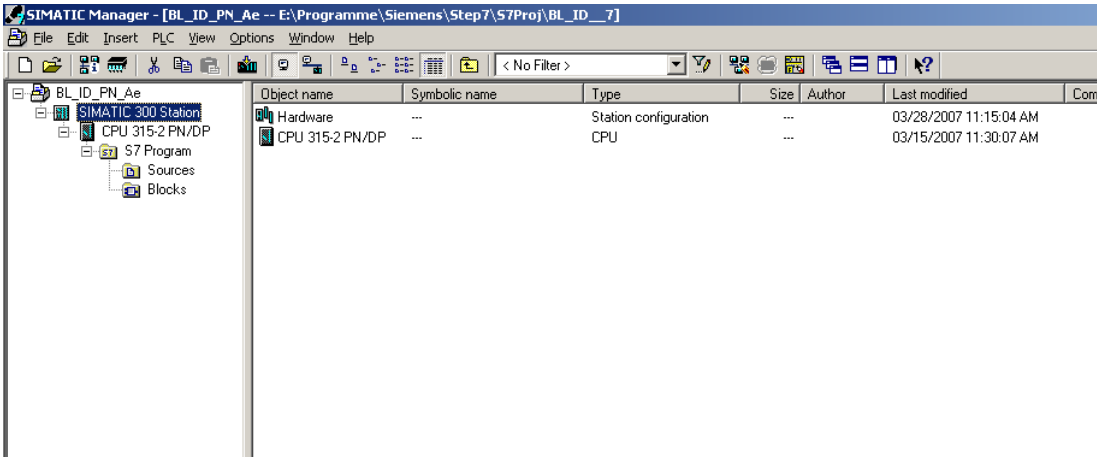
(download > configuration > GSDML PROFINET)

The procedural instructions provided in this document will enable you to commission a range of different applications.

### Starting the S7 software and loading the example project

Update the device master file (GSD) if required (before or after start). Start the “SIMATIC Basis software Step 7”. Following the Start, the window "SIMATIC manager" is activated.

*Figure 3: After the start of the SIMATIC Manager*




The example project is opened with:
File > Retrieve

Select the TURCK example file from your directory:

“BL_ID_PN_Ae.zip”

The SIMATIC manager suggests a save location (target directory) for your *BL ident*® test project. You can confirm or alter this. Confirm "Open the file".

### Hardware configuration and I/O addresses

Double clicking on "SIMATIC" in the directory tree on the left side of the window calls up, amongst others, "Hardware" on the right side of the window. Here you can make any necessary adjustments to the configuration due to a different hardware setup to that of the example project. Double clicking on the PROFINET station enables you to view the hardware configuration.

You can change the I/O addresses suggested by the SIMATIC manager. It is advantageous if you maintain the suggested I and O addresses "2..5", for this example.

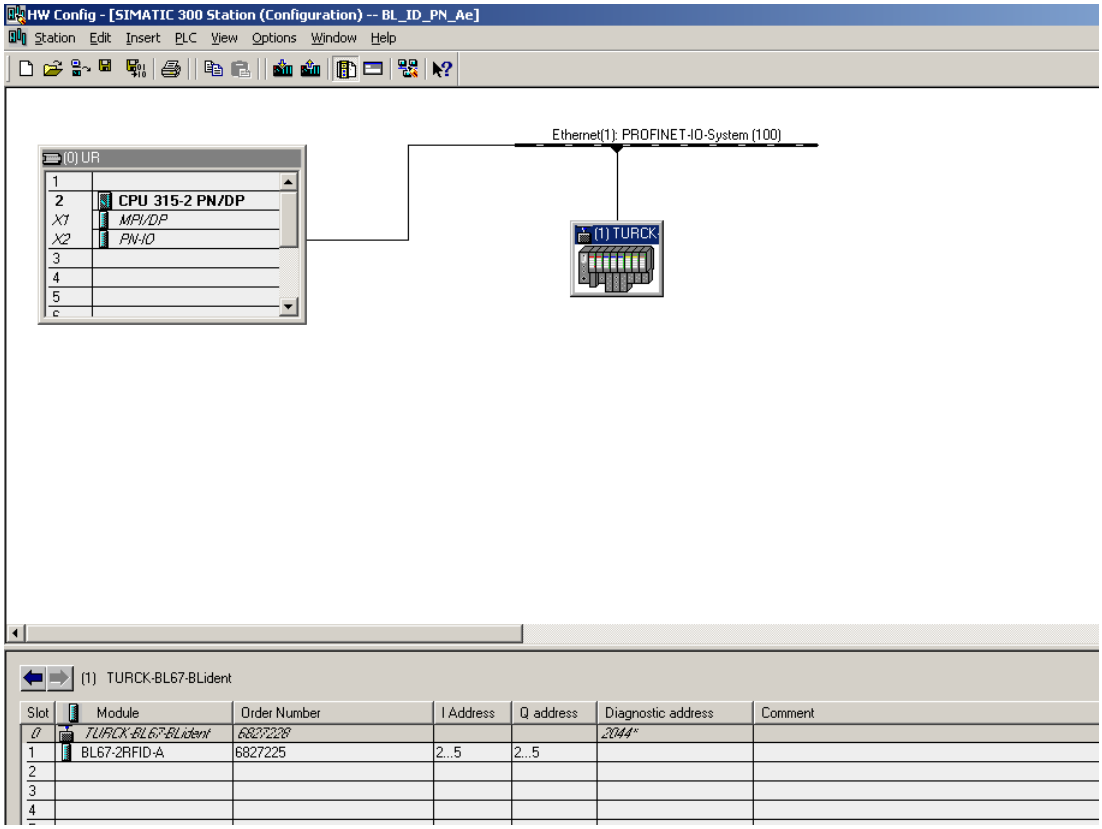
**Note**

When using the function block PIB it is necessary to choose the same value for input and output addresses.

Transfer the configuration data to the automation system (PLC > Download.)

Confirm the request for the module to be restarted.

*Figure 4: Hardware configurator*




Please observe, that the device name of your PROFINET BL67 station is stored in the gateway. Doubleclick on the PROFINET station to assign a device name to a PROFINET BL67 station or to change an existing device name in the online mode.

**Note**

In this example project the device name is „TURCK-BL67-BLident“. Please adjust a divergent device name accordingly in order to enable communication.

### Setting up the PIB function block

The main settings in the example project have already been made.

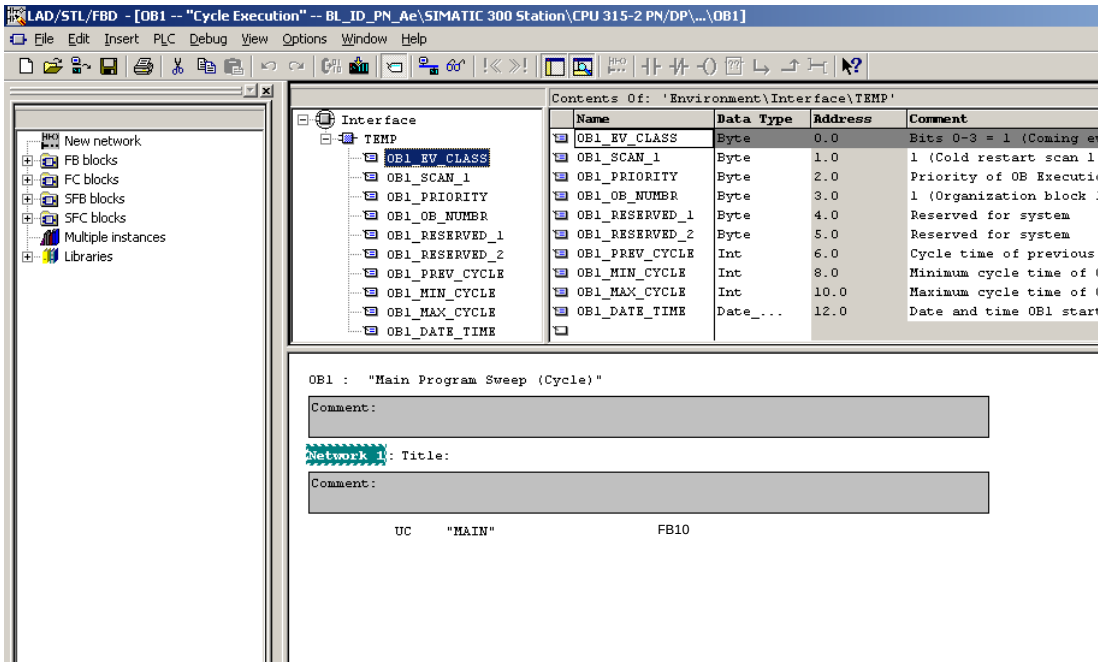
The following explanations serves to improve general understanding so that you will also be able to commission applications which differ from that of this example project.

Close the hardware configurator if it is still open.

Open the folder "Blocks" in the project tree in the left window (last point in the project tree). The block OB1 represents the top program level which is cyclically processed by the CPU.

By double clicking on OB1 you can view the program structure.

*Figure 5: Main program*




The main program OB1 calls up in principle the FB10. Close OB1 and double click in the block folder on FB10.

### PIB variable table with the FB10

The FB 10 assigns the variables according to specification (formal parameters) to the variables for the PIB instance of a channel (actual parameter).

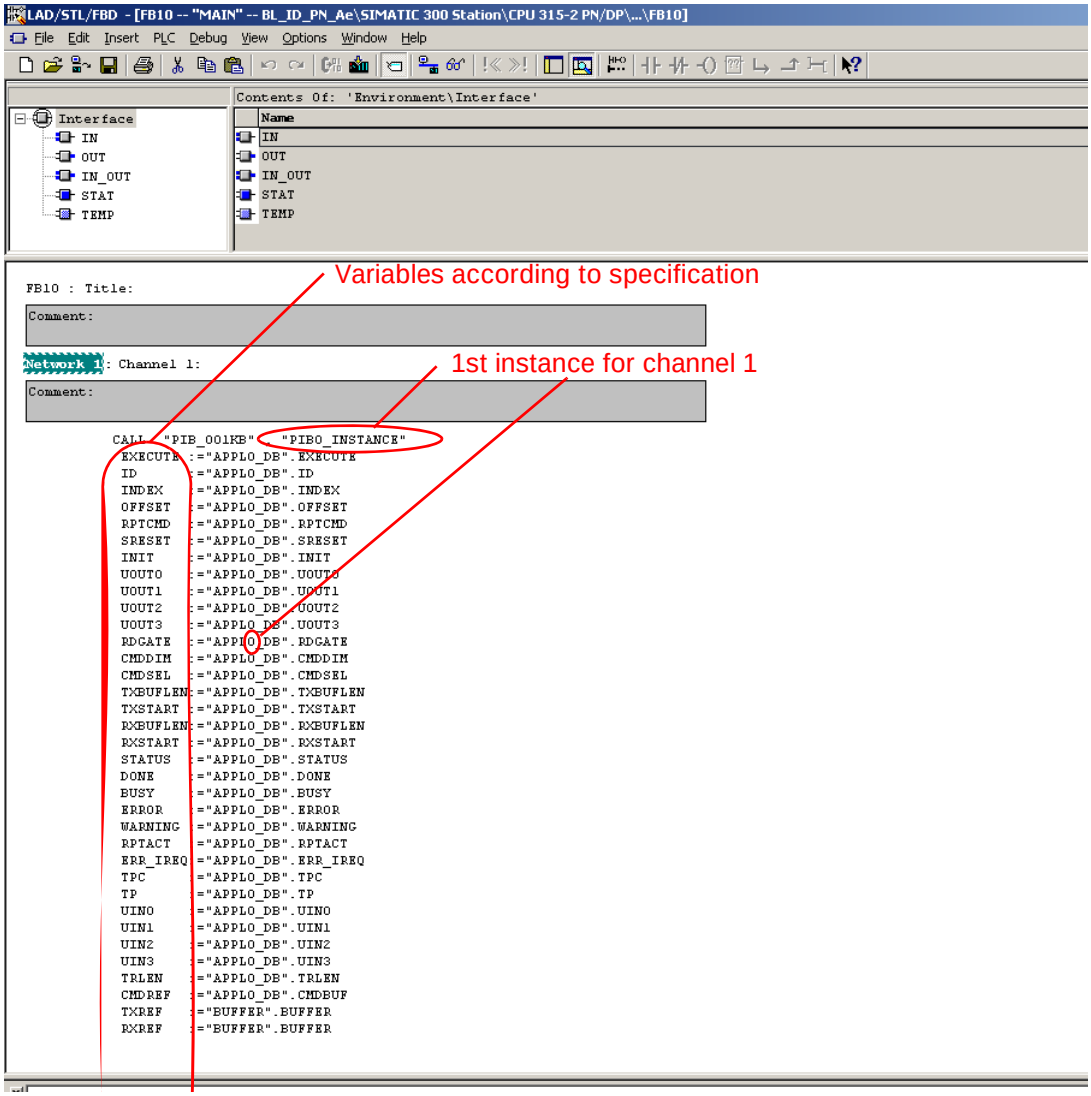
An explanation of all the variables carried out in this block can be found in „3 Definition of the Proxy Ident Block (PIB)" page 3-6.

As 2 channels for a *BL ident*® communication are available in the TURCK example project, two "instances" of the Proxy Ident Block are generated.

The PIB instance for the 1st channel is labelled "0". All the variables for the 1st instance also contain a "0".

The 2nd channel is accordingly "1" etc.

*Figure 6: Variables for the 1st instance*



### Monitoring and controlling with the variable table vartable_pibX

Close the FB10 and open via the block folder the variable table vartable_pib0. This table belongs to the 1st instance of the PIB and thus to channel 1.

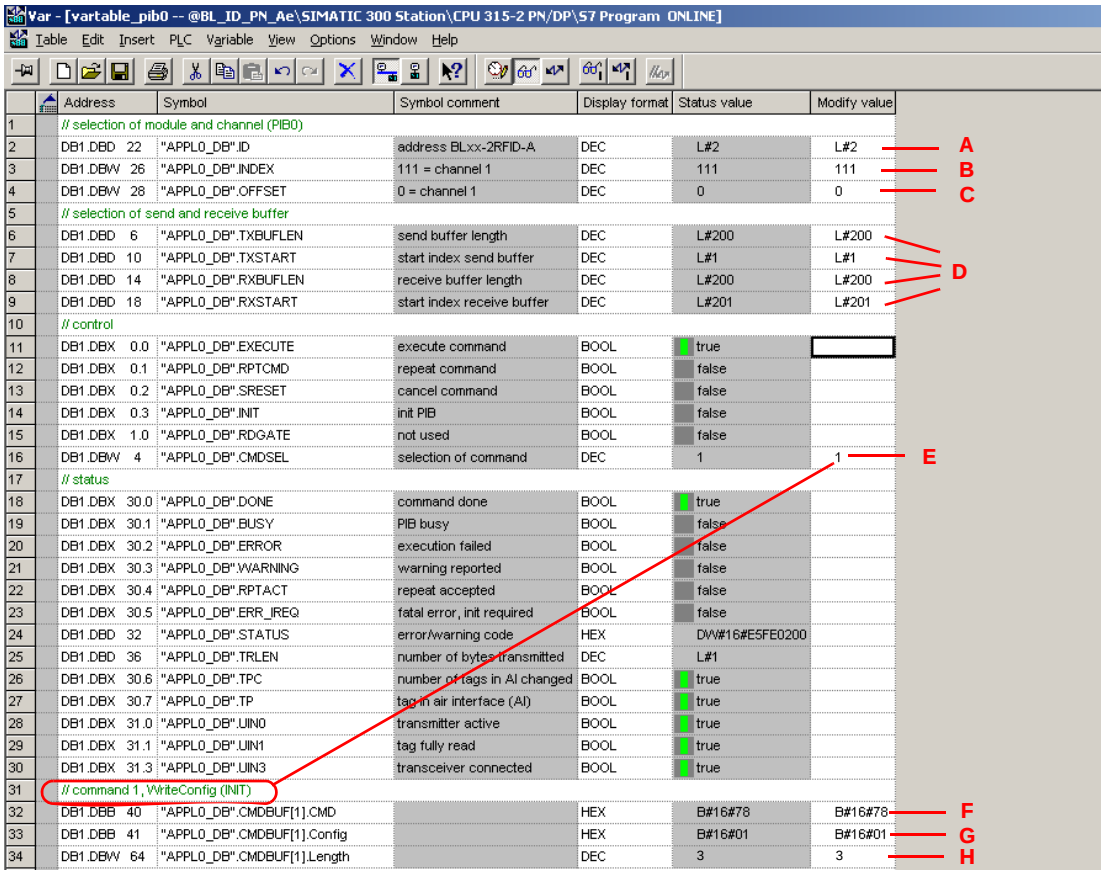
Activate the online connection to your control (PLC > Connect to > Direct CPU) to read the status values and load the control variables. The “RUN” mode will be marked green at the bottom right of the window.

Adjust the values indicated in the points A to D of the explanatory text in the column control values, should your application differ from that of the example project.

**Note**

Load the values into your control (Variable modify) and check using the column status values (Variable monitor) , that the control has accepted the values!!

*Figure 7: Entries in the vartable_pib0*




| | Address | Symbol | Symbol comment | Display format | Status value | Modify value |
|---|---|---|---|---|---|---|
| 1 | // selection of module and channel (PIB0) | | | | | |
| 2 | DB1.DBD 22 | "APPL0_DB".ID | address BLxx-2RFID-A | DEC | L#2 | L#2 |
| 3 | DB1.DBW 26 | "APPL0_DB".INDEX | 111 = channel 1 | DEC | 111 | 111 |
| 4 | DB1.DBW 28 | "APPL0_DB".OFFSET | 0 = channel 1 | DEC | 0 | 0 |
| 5 | // selection of send and receive buffer | | | | | |
| 6 | DB1.DBD 6 | "APPL0_DB".TXBUFLEN | send buffer length | DEC | L#200 | L#200 |
| 7 | DB1.DBD 10 | "APPL0_DB".TXSTART | start index send buffer | DEC | L#1 | L#1 |
| 8 | DB1.DBD 14 | "APPL0_DB".RXBUFLEN | receive buffer length | DEC | L#200 | L#200 |
| 9 | DB1.DBD 18 | "APPL0_DB".RXSTART | start index receive buffer | DEC | L#201 | L#201 |
| 10 | // control | | | | | |
| 11 | DB1.DBX 0.0 | "APPL0_DB".EXECUTE | execute command | BOOL | true | |
| 12 | DB1.DBX 0.1 | "APPL0_DB".RPTCMD | repeat command | BOOL | false | |
| 13 | DB1.DBX 0.2 | "APPL0_DB".SRESET | cancel command | BOOL | false | |
| 14 | DB1.DBX 0.3 | "APPL0_DB".INIT | init PIB | BOOL | false | |
| 15 | DB1.DBX 1.0 | "APPL0_DB".RDGATE | not used | BOOL | false | |
| 16 | DB1.DBW 4 | "APPL0_DB".CMDSEL | selection of command | DEC | 1 | 1 |
| 17 | // status | | | | | |
| 18 | DB1.DBX 30.0 | "APPL0_DB".DONE | command done | BOOL | true | |
| 19 | DB1.DBX 30.1 | "APPL0_DB".BUSY | PIB busy | BOOL | false | |
| 20 | DB1.DBX 30.2 | "APPL0_DB".ERROR | execution failed | BOOL | false | |
| 21 | DB1.DBX 30.3 | "APPL0_DB".WARNING | warning reported | BOOL | false | |
| 22 | DB1.DBX 30.4 | "APPL0_DB".RPTACT | repeat accepted | BOOL | false | |
| 23 | DB1.DBX 30.5 | "APPL0_DB".ERR_IREQ | fatal error, init required | BOOL | false | |
| 24 | DB1.DBD 32 | "APPL0_DB".STATUS | error/warning code | HEX | DW#16#E5FE0200 | |
| 25 | DB1.DBD 36 | "APPL0_DB".TRLEN | number of bytes transmitted | DEC | L#1 | |
| 26 | DB1.DBX 30.6 | "APPL0_DB".TPC | number of tags in AI changed | BOOL | true | |
| 27 | DB1.DBX 30.7 | "APPL0_DB".TP | tag in air interface (AI) | BOOL | true | |
| 28 | DB1.DBX 31.0 | "APPL0_DB".UIN0 | transmitter active | BOOL | true | |
| 29 | DB1.DBX 31.1 | "APPL0_DB".UIN1 | tag fully read | BOOL | true | |
| 30 | DB1.DBX 31.3 | "APPL0_DB".UIN3 | transceiver connected | BOOL | true | |
| 31 | // command 1, WriteConfig (INIT) | | | | | |
| 32 | DB1.DBB 40 | "APPL0_DB".CMDBUF[1].CMD | | HEX | B#16#78 | B#16#78 |
| 33 | DB1.DBB 41 | "APPL0_DB".CMDBUF[1].Config | | HEX | B#16#01 | B#16#01 |
| 34 | DB1.DBW 64 | "APPL0_DB".CMDBUF[1].Length | | DEC | 3 | 3 |

**A** *This is the starting address for the BL ident® process data of first module. The “ID” (start address) for a second module must be “6”, for a third “10” and a fourth “14”. The address for each individual channel is first of all defined with offset (legend point* **C***.*

**B** *The index “111” defines that the next data transfer undertaken (including parameter data) is to channel 1. The index “112” refers to channel 2. This applies fore every BL ident® modul of the station. These indexes are employed universally (including parameter data transfer). The indexes 101 to 108, which according to the specification are to be selected for parameter data transfer, are no longer employed.*

**C** *This offset is added to the start address (***A***). The calculated address relates to the process data of a channel. Here the offset is "0" as vartable_pib0 belongs to the first channel. The process data for a BL ident® channel is 2 bytes. The corresponding offset in vartable_pib1, which belongs to the 2nd channel is "2".*

**D** *Here the buffer area for the read and write data for the 1st instance (1st channel) is given. The „Memory requirements for the read and write data” page 2-2 is, with the selection of a PIB-1KB, limited to a total of 1 kilobyte. Here it is indicated that the send buffer for the 1st channel occupies the area 1 to 200. The receive buffer occupies the area 201 to 400.*
*For the 2nd channel the areas 401 to 600 and 600 to 800 are occupied accordingly (vartable_pib1). The option was abandoned here in the event of enough skew to use memory from different channels or for alternate read and write.*

**E** *Here 1, 2 or 3 can be entered, when only one from 3 possible commands (see vartable_pibX: command 1, WriteConfig (INIT), command 2, e.g. Inventory...) is to be executed. As only the Write-Config command („Write Config” page 3-26) is to be executed, "1" has already been entered here.*

**F** *The hexadecimal coding for the command "Write-Config" is $78_{hex}$.*

**G** *Changes to the configuration data can be made according to the specification („Config” page 3-26) by means of a reset ($01_{hex}$)(as in the example),*
*or*
*with a combination of reset and new configuration data ($03_{hex}$).*

**H** *The number of configuration data that is to be written.*
*(Here there are 3 configuration data elements for the data carrier, which will be described in more detail in the next section.)*

### On/off switching of the read/write head via configuration data

Entering the values in E to H (Figure 7:, page 2-8) completes the preparations for the sending of configuration values.

The configuration values in the TURCK example project are already in the "transmit data field" of vartable_pib0. The transmit data field is already below the command area of vartable_pibX (here X = 0).

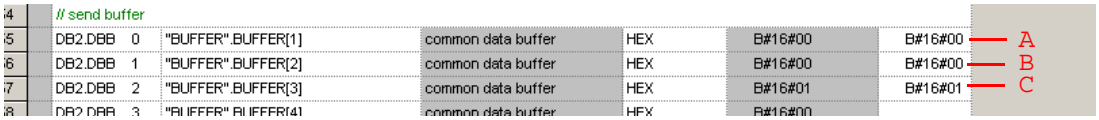
Only the value of the third configuration byte is of significance. Switch on and off of the read-write heads is requested via this value and undertaken with the next initialization. At initialisation the command „Write Config" page 2-22 is executed.

After an already successful initialisation switching on and off of the read-write heads with the command „Write Config" page 2-22 (0x78) and a "rising edge" of the control variables "APPL0_DB".EXECUTE is undertaken. To execute the command proceed as when writing and reading (e.g. „Writing on the data carrier / channel 1" page 2-14).

**Note**

Load the values into your control (Variable modify) and check using the column status values (Variable monitor) , that the control has accepted the values!!

*Figure 8: Transmission data field of vartable_pib0*




**A** *and*

**B** *These data fields need not to be adjusted. Originally these fields were used for entries regarding number of blocks and number of bytes per block. Keep the value $00_{hex}$. The values of the data carrier currently applied are internally read and processed.*

**C** *With the value $01_{hex}$ the transmitter (antenna )of the read/write head is turned on. With the value $00_{hex}$ you can turn off the transmitter.*

### Initialisation of the channel 1

In the section „Setting up the PIB function block” page 2-6 you have been familiarised with the relevant settings (control values) for an initialisation. If your *BL ident*®project diverges from the example project, you have adjusted the control values.

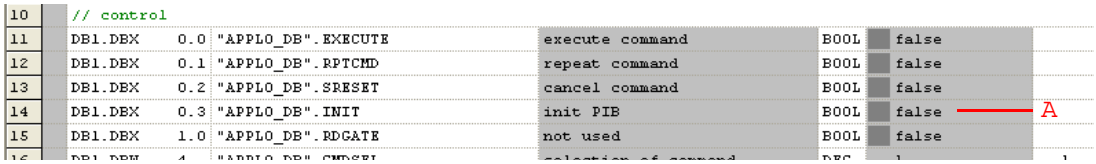
**Note**

Load all the values that have been described in the previous sections into your control (Variable modify) ! Check via the status values column (Variable monitor) , that the control has accepted the values!

Now conduct the initialisation. Ensure that the online connection to your control is active. The “RUN” mode will be marked green at the bottom right of the window.

With a "rising signal edge" of the control variable “APPL0_DB“.INIT the command "Initialisation" is executed. You generate the positive edge by setting the variable from “false” to “true”. Enter either "1" or "true" as the control value.

*Figure 9: Control field of vartable_pib0*



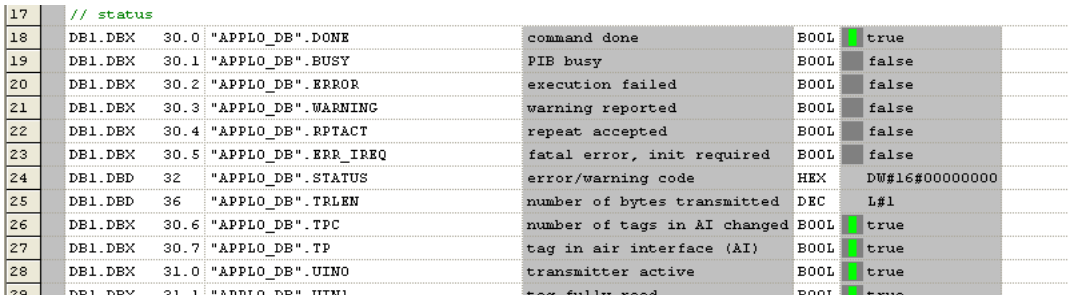

**A** *The initialisation is carried out with the rising signal edge (change from false-> true or 0->1)*

With:
Variable > modify or

the command "Initialisation" is executed.

You can view the execution of the command in the status field of vartable_pib0.

*Figure 10: Status field of vartable_pib0*



| | | | | | | |
|---|---|---|---|---|---|---|
| 17 | | // status | | | | |
| 18 | | DB1.DBX | 30.0 | "APPL0_DB".DONE | command done | BOOL | true |
| 19 | | DB1.DBX | 30.1 | "APPL0_DB".BUSY | PIB busy | BOOL | false |
| 20 | | DB1.DBX | 30.2 | "APPL0_DB".ERROR | execution failed | BOOL | false |
| 21 | | DB1.DBX | 30.3 | "APPL0_DB".WARNING | warning reported | BOOL | false |
| 22 | | DB1.DBX | 30.4 | "APPL0_DB".RPTACT | repeat accepted | BOOL | false |
| 23 | | DB1.DBX | 30.5 | "APPL0_DB".ERR_IREQ | fatal error, init required | BOOL | false |
| 24 | | DB1.DBD | 32 | "APPL0_DB".STATUS | error/warning code | HEX | DW#16#00000000 |
| 25 | | DB1.DBD | 36 | "APPL0_DB".TRLEN | number of bytes transmitted | DEC | L#1 |
| 26 | | DB1.DBX | 30.6 | "APPL0_DB".TPC | number of tags in AI changed | BOOL | true |
| 27 | | DB1.DBX | 30.7 | "APPL0_DB".TP | tag in air interface (AI) | BOOL | true |
| 28 | | DB1.DBX | 31.0 | "APPL0_DB".UINO | transmitter active | BOOL | true |

The status variable “APPL0_DB“.DONE changes momentarily to the condition "Busy" and then returns to indicating "Command executed"= "true". The error free execution is confirmed by “APPL0_DB“.ERROR = false.

You can find explanations of a number of the error codes for the status variables “APPL0_DE”.STATUS, in particular those for the *BL ident*® specific errors in „Warnings and error messages” page 2-26.

A complete description of the status data can be found in „Warnings and error messages” page 2-26 and„3.1.2 Errors and warnings” page 3-13.

Reset the variable “APPL0_DB”.INIT to “false”, if initialisation has been successful.

With:
Variable > modify or

"false" becomes the status value again.

### Reading of the UID from data carrier / channel 1

Every RFID data carrier receives a „UID” page 4-6 (unique identifier) in the factory. The UID is a unique TAG identification number world-wide and consists of 8 bytes. See „Access to the data range of the data carrier” page 2-33.

Reading of the UID is executed with the command “Inventory” . The instruction code $69_{hex}$ for the inventory is already entered into the field “Command 2” of vartable_pib0. A comprehensive description of this command code can be found at „Inventory” page 3-29.

*Figure 11: Command 2 of vartable_pib0*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 35 | | // command 2, i.e. Inventory | | | | | |
| 36 | DB1.DBB | 78 | "APPL0_DB".CMDBUF[2].CMD | | HEX | B#16#69 | B#16#69 |
| 37 | DB1.DBW | 102 | "APPL0_DB".CMDBUF[2].Length | | DEC | 0 | |
| 38 | DB1.DBD | 104 | "APPL0_DB".CMDBUF[2].StartAddress | | DEC | L#0 | |

Ensure that the online connection to your control is active. The “RUN” mode will be marked green at the bottom right of the window.

The control value “APPL0_DP“.CMDSEL of the vartable_pib0 is most likely to still have the value "1" as you have just carried out the "initialisation" with the command 1.

Enter "2" for this control value in order to select command 2.

*Figure 12: Control field of vartable_pib0*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 10 | | // control | | | | | |
| 11 | DB1.DBX | 0.0 | "APPL0_DB".EXECUTE | execute command | BOOL | false | |
| 12 | DB1.DBX | 0.1 | "APPL0_DB".RPTCMD | repeat command | BOOL | false | |
| 13 | DB1.DBX | 0.2 | "APPL0_DB".SRESET | cancel command | BOOL | false | |
| 14 | DB1.DBX | 0.3 | "APPL0_DB".INIT | init PIB | BOOL | false | |
| 15 | DB1.DBX | 1.0 | "APPL0_DB".RDGATE | not used | BOOL | false | |
| 16 | DB1.DBW | 4 | "APPL0_DB".CMDSEL | selection of command | DEC | 2 | 2 |

**Note**

Load the **all** values into your control (Variable modify) and check using the column status values (Variable monitor) , that the control has accepted the values!

With a "rising signal edge" of the control variable “APPL0_DB“.EXECUTE the command "Inventory" is shown. You generate the positive edge by setting the variable from “false” to “true”. Enter either "1" or "true" as the control value.

*Figure 13: Control field of vartable_pib0*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 10 | | // control | | | | | |
| 11 | DB1.DBX | 0.0 | "APPL0_DB".EXECUTE | execute command | BOOL | false | A |
| 12 | DB1.DBX | 0.1 | "APPL0_DB".RPTCMD | repeat command | BOOL | false | |
| 13 | DB1.DBX | 0.2 | "APPL0_DB".SRESET | cancel command | BOOL | false | |

**A** *“Inventory” is carried out with the rising signal edge (change from false-> true or 0->1)*
With:
Variable > modify or

the command "Inventory" is executed.

You can view the execution of the command in the status field of vartable_pib0.

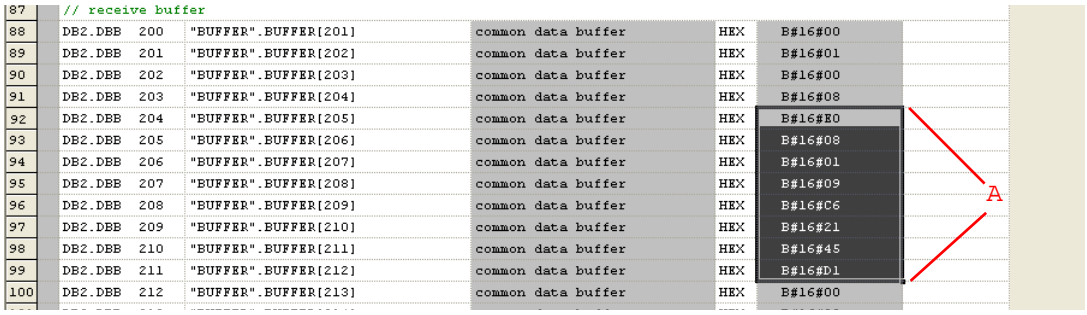
The status variable “APPL0_DB“.DONE changes momentarily to the condition "Busy" and then returns to indicating "Command executed"= "true". The error free execution is confirmed by “APPL0_DB“.ERROR = false.

A total of 8 UID data bytes are received with "Inventory". The size of the data transmitted here is “APPL0_DB”.TRLEN =“12”.

A complete description of the status data can be found in „Warnings and error messages” page 2-26 and „3.1.2 Errors and warnings” page 3-13

You can read the “Unique Identifier / UID” in the received data field of the vartable_pib0.

*Figure 14: The UID in the received data field*




| | | | | | | |
|---|---|---|---|---|---|---|
| 87 | // receive buffer | | | | | |
| 88 | DB2.DBB | 200 | "BUFFER".BUFFER[201] | common data buffer | HEX | B#16#00 |
| 89 | DB2.DBB | 201 | "BUFFER".BUFFER[202] | common data buffer | HEX | B#16#01 |
| 90 | DB2.DBB | 202 | "BUFFER".BUFFER[203] | common data buffer | HEX | B#16#00 |
| 91 | DB2.DBB | 203 | "BUFFER".BUFFER[204] | common data buffer | HEX | B#16#08 |
| 92 | DB2.DBB | 204 | "BUFFER".BUFFER[205] | common data buffer | HEX | B#16#E0 |
| 93 | DB2.DBB | 205 | "BUFFER".BUFFER[206] | common data buffer | HEX | B#16#08 |
| 94 | DB2.DBB | 206 | "BUFFER".BUFFER[207] | common data buffer | HEX | B#16#01 |
| 95 | DB2.DBB | 207 | "BUFFER".BUFFER[208] | common data buffer | HEX | B#16#09 |
| 96 | DB2.DBB | 208 | "BUFFER".BUFFER[209] | common data buffer | HEX | B#16#C6 |
| 97 | DB2.DBB | 209 | "BUFFER".BUFFER[210] | common data buffer | HEX | B#16#21 |
| 98 | DB2.DBB | 210 | "BUFFER".BUFFER[211] | common data buffer | HEX | B#16#45 |
| 99 | DB2.DBB | 211 | "BUFFER".BUFFER[212] | common data buffer | HEX | B#16#D1 |
| 100 | DB2.DBB | 212 | "BUFFER".BUFFER[213] | common data buffer | HEX | B#16#00 |

**A** *The 8 byte UID can be found from byte 5 (Receive Buffer 5) in the receive data buffer. Byte 5 reflects the MSB and byte 12 of the LSB of the UID. Byte 1 and 2 always show for BL ident® 0001$_{hex}$. Bulk read capability systems indicate here which of the data carriers have been read from the current “bulk”. Byte 3 and 4 together indicate the length of the UID (here: 0008$_{hex}$,.....*

Reset the variable “APPL0_DB“.EXECUTE to „false“, when the UID has been successfully read.

With:
Variable > modify or

"false" becomes the status value again.

### Writing on the data carrier / channel 1

In this section writing of 32 bytes of data with any content to your RFID data carrier are explained.

Reading of the data carrier of the 1st channel is possible if the „Initialisation of the channel 1” page 2-11 has been implemented.

In this example we have selected data which are easy to recognise again in „Read out data carrier / channel 1” page 2-17.

The writing of the data is implemented with the command “Physical_Write” . Enter the command code 71$_{hex}$ for the Physical_Write in the field "command 3" of vartable_pib0.. A comprehensive description of this command code can be found at „Physical_Write” page 3-25.

Ensure that the online connection to your control is active. The “RUN” mode will be marked green at the bottom right of the window.

Enter "3" for the control value “APPL0_DP“.CMDSEL of vartable_pib0 in order to select the command 3.

In the transmit data field enter the 32 bytes as a hexadecimal number. Following the writing, we will explain the reading. The following numerical sequence:
1, 2, 3, 4, 5, 6, 7, 8, 9, A, B... 20, which is easy to remember, is entered.

You can enter the hexadecimal numbers either single or double digit and without any further format details (B#16#..). The SIMATIC manager converts them to the correct format.

*Figure 15: Preparing the vartable_pib0 for writing*

Var - [vartable_pib0 -- @BL_ID_PN_Ae\SIMATIC 300 Station\CPU 315-2 PN/DP\S7 Program ONLINE]

Table Edit Insert PLC Variable View Options Window Help

| | Address | Symbol | Symbol comment | Display format | Status value | Modify value | |
|---|---|---|---|---|---|---|---|
| 38 | DB1.DBD 104 | "APPL0_DB".CMDBUF[2].StartAddress | | DEC | L#0 | | |
| 39 | DB1.DBB 108 | "APPL0_DB".CMDBUF[2].Attributes | | HEX | B#16#00 | | |
| 40 | // command 3, ... | | | | | | |
| 41 | DB1.DBB 116 | "APPL0_DB".CMDBUF[3].CMD | | HEX | B#16#71 | B#16#71 | A |
| 42 | DB1.DBB 120 | "APPL0_DB".CMDBUF[3].UID[1] | | HEX | B#16#00 | B#16#00 | B |
| 43 | DB1.DBB 121 | "APPL0_DB".CMDBUF[3].UID[2] | | HEX | B#16#00 | B#16#00 | |
| 44 | DB1.DBB 122 | "APPL0_DB".CMDBUF[3].UID[3] | | HEX | B#16#00 | B#16#00 | |
| 45 | DB1.DBB 123 | "APPL0_DB".CMDBUF[3].UID[4] | | HEX | B#16#00 | | |
| 46 | DB1.DBB 124 | "APPL0_DB".CMDBUF[3].UID[5] | | HEX | B#16#00 | | |
| 47 | DB1.DBB 125 | "APPL0_DB".CMDBUF[3].UID[6] | | HEX | B#16#00 | | |
| 48 | DB1.DBB 126 | "APPL0_DB".CMDBUF[3].UID[7] | | HEX | B#16#00 | | |
| 49 | DB1.DBB 127 | "APPL0_DB".CMDBUF[3].UID[8] | | HEX | B#16#00 | | |
| 50 | DB1.DBW 140 | "APPL0_DB".CMDBUF[3].Length | | DEC | 32 | 32 | C |
| 51 | DB1.DBD 142 | "APPL0_DB".CMDBUF[3].StartAddress | | DEC | L#0 | L#0 | D |
| 52 | DB1.DBB 146 | "APPL0_DB".CMDBUF[3].Attributes | | HEX | B#16#00 | B#16#00 | E |
| 53 | DB1.DBB 147 | "APPL0_DB".CMDBUF[3].NextMode | | HEX | B#16#00 | | |
| 54 | // send buffer | | | | | | |
| 55 | DB2.DBB 0 | "BUFFER".BUFFER[1] | common data buffer | HEX | B#16#01 | B#16#01 | F |
| 56 | DB2.DBB 1 | "BUFFER".BUFFER[2] | common data buffer | HEX | B#16#02 | B#16#02 | |
| 57 | DB2.DBB 2 | "BUFFER".BUFFER[3] | common data buffer | HEX | B#16#03 | B#16#03 | |
| 58 | DB2.DBB 3 | "BUFFER".BUFFER[4] | common data buffer | HEX | B#16#04 | B#16#04 | |
| 59 | DB2.DBB 4 | "BUFFER".BUFFER[5] | common data buffer | HEX | B#16#05 | B#16#05 | |
| 60 | DB2.DBB 5 | "BUFFER".BUFFER[6] | common data buffer | HEX | B#16#06 | B#16#06 | |
| 61 | DB2.DBB 6 | "BUFFER".BUFFER[7] | common data buffer | HEX | B#16#07 | B#16#07 | |
| 62 | DB2.DBB 7 | "BUFFER".BUFFER[8] | common data buffer | HEX | B#16#08 | B#16#08 | |
| 63 | DB2.DBB 8 | "BUFFER".BUFFER[9] | common data buffer | HEX | B#16#09 | B#16#09 | |
| 64 | DB2.DBB 9 | "BUFFER".BUFFER[10] | common data buffer | HEX | B#16#0A | B#16#0A | |
| 65 | DB2.DBB 10 | "BUFFER".BUFFER[11] | common data buffer | HEX | B#16#0B | B#16#0B | |
| 66 | DB2.DBB 11 | "BUFFER".BUFFER[12] | common data buffer | HEX | B#16#0C | B#16#0C | |
| 67 | DB2.DBB 12 | "BUFFER".BUFFER[13] | common data buffer | HEX | B#16#0D | B#16#0D | |
| 68 | DB2.DBB 13 | "BUFFER".BUFFER[14] | common data buffer | HEX | B#16#0E | B#16#0E | |

**A** *The code for the command 3 which is to be carried out next,is entered here. An overview of all the possible commands can be found in „3.1.3 Commands" page 3-23.*
*The command code $71_{hex}$ stands for the order to write on a physically available TAG "Physical_Write"..*

**B** *This 8 byte data field can contain a UID. This UID is always compared to the TAG's UID when values ≠ 00 are entered here.*
*Make sure that all 8 bytes have the value "00" if you don't want to carry out the UID compare function.*

**C** *Here you enter the number of bytes that should be transmitted from the transmit data field. The number of possible bytes is dependent on the size of the transmit data field (page 2-9) and the memory capacity of the TAG employed. In this TURCK example 32 bytes are written to the TAG (112 byte). The size of the transmit data field here is 200 bytes.*

**D** *Every byte on the TAG can be explicitly addressed as the start address using this address. In this example the start address L#0 has been selected. Use another data carrier than that in this example project, observe section „User data range of the data carrier variants" page 2-33.*

**E** *These "attributes" enable a number of commands to be specified in more detail. This value is not taken into account with the read and write command.*
*In „3.1.3 Commands" page 3-23 you will find an overview of all possible commands and the options available for specifying these in more detail via "attributes". Please also take note of the following „Definitions in the command and diagnostics level" page 2-21.*

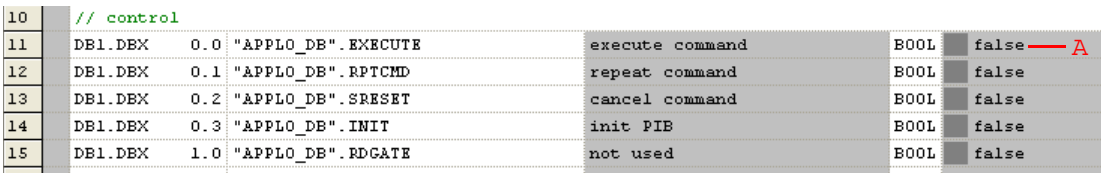
**F** *In this transmit data field you enter the data you want to write on your TAG.*

**Note**

Load all the values in the fields "control", "command 3" and "transmit data field" in your control (Variable control) and check via the status value column (Variable monitor), that the control has accepted the values!

With a "rising signal edge" of the control variable “APPL0_DB“.EXECUTE the command “Physical_Write” is executed. You generate the positive edge by setting the variable from “false” to “true”. Enter either "1" or "true" as the control value.

*Figure 16: Control field of vartable_pib0*



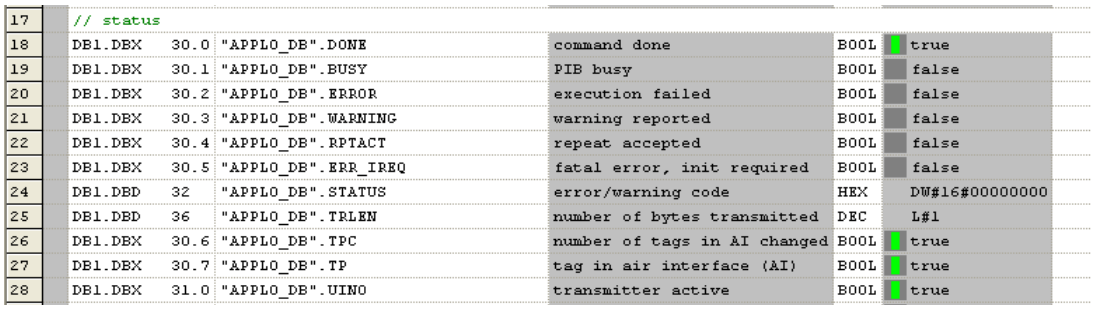

**A** *"Physical_Wríte" proceeds with the rising signal edge (change from false-> true or 0->1)*

With:
Variable > modify or

the command "Physical_Write" is executed..

You can view the execution of the command in the status field of vartable_pib0.

*Figure 17: Status field of vartable_pib0*




The status variable “APPL0_DB“.DONE changes momentarily to the condition "Busy" and then returns to indicating "Command executed"= "true". The error free execution is confirmed by “APPL0_DB“.ERROR = false.

A complete description of the status data can be found in „Warnings and error messages” page 2-26 and „3.1.2 Errors and warnings” page 3-13

Reset the variable “APPL0_DB“.EXECUTE to “false”, if writing has been successful.

With:
Variable > modify or

"false" becomes the status value again.

### Read out data carrier / channel 1

In this section reading of 32 bytes of data with any content to your RFID data carrier are explained.

Reading of the data carrier of the 1st channel is possible if the „Initialisation of the channel 1” page 2-11 has been implemented.

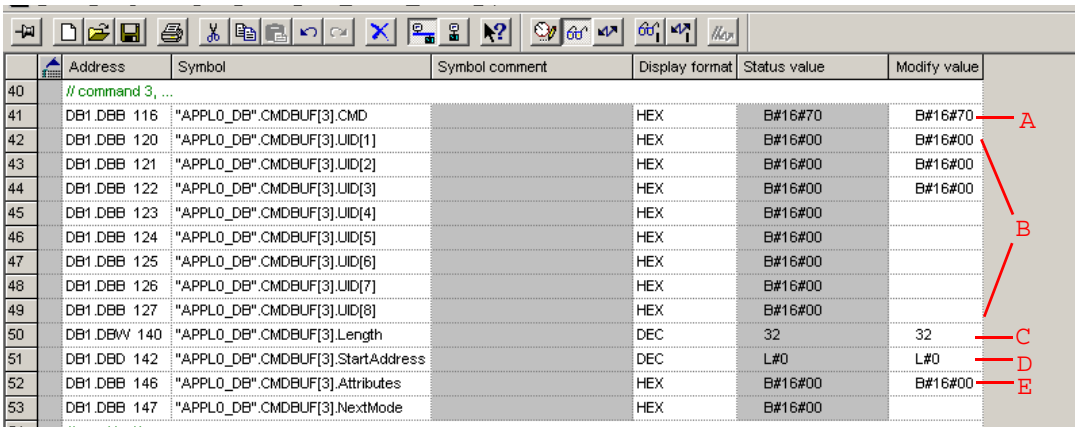
In the previous section you have written data (arbitrary) to the data carrier using the command "Physical_Write". In this section the same data will now be read from the data carrier using the command "Physical_Read" .

Enter the command code $70_{hex}$ for the Physical_Read in the field "command 3" of vartable_pib0. A comprehensive description of this command code can be found at „Physical_Read” page 3-24.

Ensure that the online connection to your control is active. The “RUN” mode will be marked green at the bottom right of the window.

Enter "3" for the control value “APPL0_DP“.CMDSEL of vartable_pib0 in order to select the command 3.

*Figure 18: Preparing the vartable_pib0 for reading*



| | Address | Symbol | Symbol comment | Display format | Status value | Modify value | |
|---|---|---|---|---|---|---|---|
| 40 | // command 3, ... | | | | | | |
| 41 | DB1.DBB 116 | "APPL0_DB".CMDBUF[3].CMD | | HEX | B#16#70 | B#16#70 | A |
| 42 | DB1.DBB 120 | "APPL0_DB".CMDBUF[3].UID[1] | | HEX | B#16#00 | B#16#00 | B |
| 43 | DB1.DBB 121 | "APPL0_DB".CMDBUF[3].UID[2] | | HEX | B#16#00 | B#16#00 | |
| 44 | DB1.DBB 122 | "APPL0_DB".CMDBUF[3].UID[3] | | HEX | B#16#00 | B#16#00 | |
| 45 | DB1.DBB 123 | "APPL0_DB".CMDBUF[3].UID[4] | | HEX | B#16#00 | | |
| 46 | DB1.DBB 124 | "APPL0_DB".CMDBUF[3].UID[5] | | HEX | B#16#00 | | |
| 47 | DB1.DBB 125 | "APPL0_DB".CMDBUF[3].UID[6] | | HEX | B#16#00 | | |
| 48 | DB1.DBB 126 | "APPL0_DB".CMDBUF[3].UID[7] | | HEX | B#16#00 | | |
| 49 | DB1.DBB 127 | "APPL0_DB".CMDBUF[3].UID[8] | | HEX | B#16#00 | | |
| 50 | DB1.DBW 140 | "APPL0_DB".CMDBUF[3].Length | | DEC | 32 | 32 | C |
| 51 | DB1.DBD 142 | "APPL0_DB".CMDBUF[3].StartAddress | | DEC | L#0 | L#0 | D |
| 52 | DB1.DBB 146 | "APPL0_DB".CMDBUF[3].Attributes | | HEX | B#16#00 | B#16#00 | E |
| 53 | DB1.DBB 147 | "APPL0_DB".CMDBUF[3].NextMode | | HEX | B#16#00 | | |

**A** *The code for the command 3 which is to be carried out next,is entered here. An overview of all the possible commands can be found in „3.1.3 Commands” page 3-23.*
*The command code $70_{hex}$ stands for the order to write on a physically available TAG “Physical_Read”..*

**B** *This 8 byte data field can contain a UID. This UID is always compared to the TAG's UID when values ≠ 00 are entered here.*
*Make sure that all 8 bytes have the value "00" if you don't want to carry out the UID compare function.*

**C** *Here you enter the number of bytes that should be transmitted to the receive data field. The number of possible bytes is dependent on the size of the received data field (page 2-9) and the memory capacity of the TAG employed. In this TURCK example 32 bytes are read from the TAG (112 byte). The size of the received data field here is 200 bytes.*

**D** *Every byte on the TAG can be explicitly addressed as the start address using this address. In this example the start address L#0 has been selected. If you use a different data carrier as in theis example project observe section „User data range of the data carrier variants” page 2-33.*

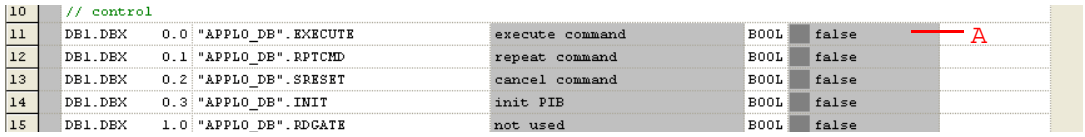
**E** *These "attributes" enable a number of commands to be specified in more detail. This value is not taken into account with the read and write command.*
*In „3.1.3 Commands” page 3-23 you will find an overview of all possible commands and the options available for specifying these in more detail via "attributes". Please also take note of the following „Definitions in the command and diagnostics level” page 2-21.*

**Note**

Load all the values in the fields "control" and "command 3" in your control (Variable control) and check via the status value column (Variable monitor) , that the control has accepted the values!

With a "rising signal edge" of the control variable “APPL0_DB“.EXECUTE the command “Physical_Read” is executed. You generate the positive edge by setting the variable from “false” to “true”. Enter either "1" or "true" as the control value.

*Figure 19: Control field of vartable_pib0*

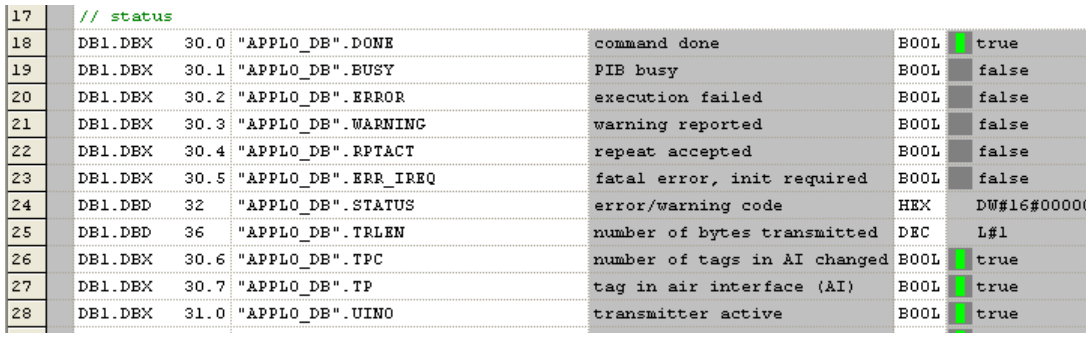



**A** *"Physical_Read" proceeds with the rising signal edge (change from false-> true or 0->1)*
With:
Variable > modify or

the command "Physical_Read" is executed.

You can view the execution of the command in the status field of vartable_pib0.

*Figure 20: Status field of vartable_pib0*




The status variable “APPL0_DB“.DONE changes momentarily to the condition "Busy" and then returns to indicating "Command executed"= "true". The error free execution is confirmed by “APPL0_DB“.ERROR = false.

The size of the data received here is “APPL0_DB”.TRLEN =“32”.

A complete description of the status data can be found in „Warnings and error messages” page 2-26 and „3.1.2 Errors and warnings” page 3-13

In the received data field with vartable_pib0 you find the read data form the TAG:

*Figure 21: Received data field of vartable_pib0 following successful reading*

| | | | | | | |
|---|---|---|---|---|---|---|
| 87 | // receive buffer | | | | | |
| 88 | DB2.DBB | 200 | "BUFFER".BUFFER[201] | common data buffer | HEX | B#16#01 |
| 89 | DB2.DBB | 201 | "BUFFER".BUFFER[202] | common data buffer | HEX | B#16#02 |
| 90 | DB2.DBB | 202 | "BUFFER".BUFFER[203] | common data buffer | HEX | B#16#03 |
| 91 | DB2.DBB | 203 | "BUFFER".BUFFER[204] | common data buffer | HEX | B#16#04 |
| 92 | DB2.DBB | 204 | "BUFFER".BUFFER[205] | common data buffer | HEX | B#16#05 |
| 93 | DB2.DBB | 205 | "BUFFER".BUFFER[206] | common data buffer | HEX | B#16#06 |
| 94 | DB2.DBB | 206 | "BUFFER".BUFFER[207] | common data buffer | HEX | B#16#07 |
| 95 | DB2.DBB | 207 | "BUFFER".BUFFER[208] | common data buffer | HEX | B#16#08 |
| 96 | DB2.DBB | 208 | "BUFFER".BUFFER[209] | common data buffer | HEX | B#16#09 |
| 97 | DB2.DBB | 209 | "BUFFER".BUFFER[210] | common data buffer | HEX | B#16#0A |
| 98 | DB2.DBB | 210 | "BUFFER".BUFFER[211] | common data buffer | HEX | B#16#0B |
| 99 | DB2.DBB | 211 | "BUFFER".BUFFER[212] | common data buffer | HEX | B#16#0C |
| 100 | DB2.DBB | 212 | "BUFFER".BUFFER[213] | common data buffer | HEX | B#16#0D |
| 101 | DB2.DBB | 213 | "BUFFER".BUFFER[214] | common data buffer | HEX | B#16#0E |
| 102 | DB2.DBB | 214 | "BUFFER".BUFFER[215] | common data buffer | HEX | B#16#0F |
| 103 | DB2.DBB | 215 | "BUFFER".BUFFER[216] | common data buffer | HEX | B#16#10 |
| 104 | DB2.DBB | 216 | "BUFFER".BUFFER[217] | common data buffer | HEX | B#16#11 |
| 105 | DB2.DBB | 217 | "BUFFER".BUFFER[218] | common data buffer | HEX | B#16#12 |
| 106 | DB2.DBB | 218 | "BUFFER".BUFFER[219] | common data buffer | HEX | B#16#13 |
| 107 | DB2.DBB | 219 | "BUFFER".BUFFER[220] | common data buffer | HEX | B#16#14 |
| 108 | DB2.DBB | 220 | "BUFFER".BUFFER[221] | common data buffer | HEX | B#16#15 |
| 109 | DB2.DBB | 221 | "BUFFER".BUFFER[222] | common data buffer | HEX | B#16#16 |
| 110 | DB2.DBB | 222 | "BUFFER".BUFFER[223] | common data buffer | HEX | B#16#17 |
| 111 | DB2.DBB | 223 | "BUFFER".BUFFER[224] | common data buffer | HEX | B#16#18 |

Reset the variable “APPL0_DB“.EXECUTE to „false“, when the reading has been successfully completed..

With:
Variable > modify or

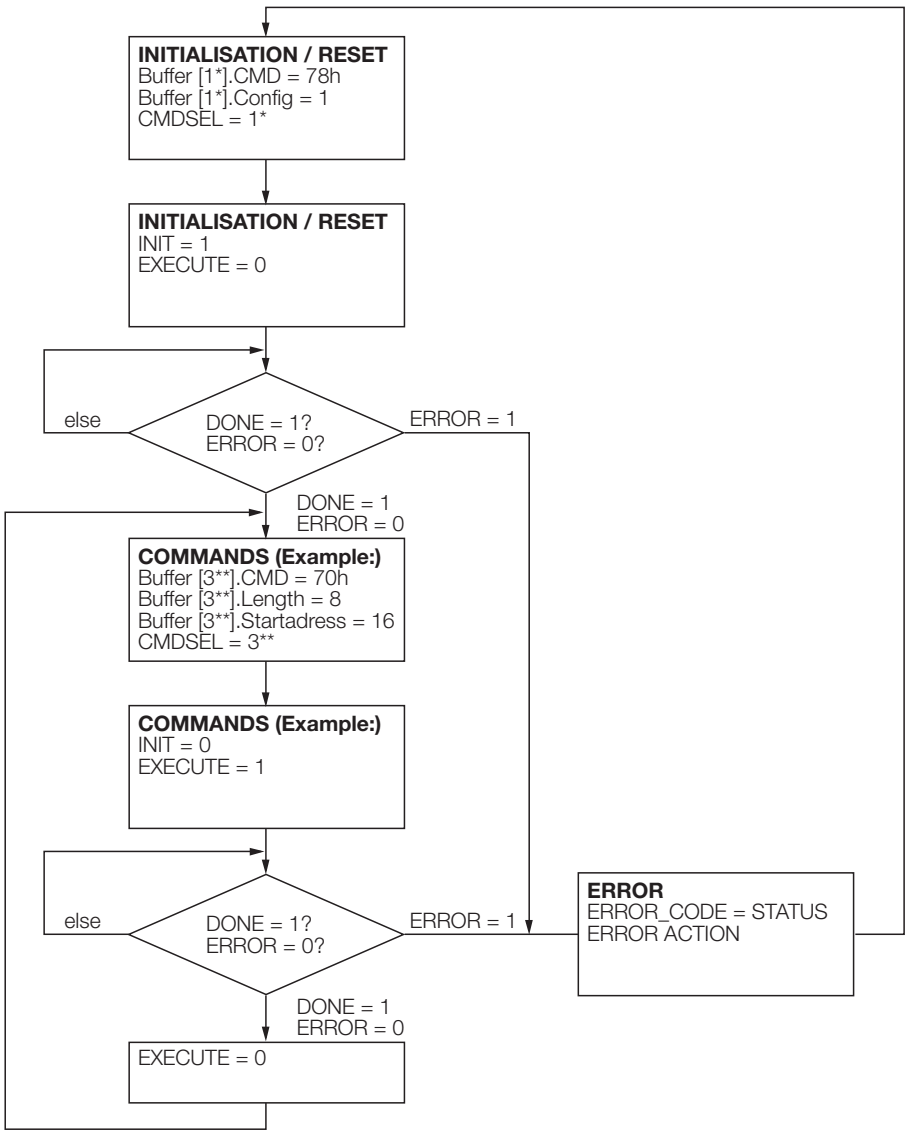
"false" becomes the status value again.

## Flow chart on the function of the PIB

The following flow chart shows the function of the PIB at a glance.

Further support for your own programming is provided by „Extract from the specification” page 3-1 and the following sections.

*Figure 22:*
*PIB flow chart*




* must be 0
** can be 3

**Note**

Please note that the query of the parameters DONE ,ERROR, TP etc. is always relevant to a signal change (edge).

## Definitions in the command and diagnostics level

The commissioning of the TURCK *BL ident*® system with the “Proxy Ident Function Block” includes some divergences to the „Extract from the specification” page 3-1 (PROFIBUS specification). The divergences concern the command and diagnostics level.

The following conformity table indicates which command and status or diagnostics messages of the complete specification *BL ident*® are not supported:

*Table 1:*

*Conformity table*

| Name | Type | Additional information about the TURCK specific version | conform? (Y/N) |
|---|---|---|---|
| To point 3.1.4 of the PROFIBUS specification | | | |
| Read | Command | | N |
| Get | Command | | Y |
| Physical_Read | Command | | Y |
| Write | Command | | N |
| Put | Command | | N |
| Physical_Write | Command | | Y |
| Format | Command | | N |
| Create | Command | | N |
| Delete | Command | | N |
| Clear | Command | | N |
| Update | Command | | N |
| Next | Command | | Y |
| Get-Directory | Command | | N |
| Set-Attribute | Command | | N |
| Get-Attribute | Command | | N |
| Write Config | Command | | Y |
| Read Config | Command | | Y |
| Mem-Status | Command | | Y |
| Dev-Status | Command | | Y |
| Inventory | Command | | Y |
| Read-Bar-Code | Command | | N |
| To point 4.2.1 of the PROFIBUS specification | | | |
| Reading-Gate | Control Bit | | N |
| Repeat-Command | Control Bit | | N |
| Soft-Reset | Control Bit | | Y |

*Table 1: (Forts.) Conformity table*

| Name | Type | Additional information about the TURCK specific version | conform? (Y/N) |
|---|---|---|---|
| To point 4.2.2 of the PROFIBUS specification | | | |
| Target_Presence_Changed | Status Bit | | Y |
| Target_Present | Status Bit | | Y |
| Soft_Reset_Active | Status Bit | | Y |
| Repeat_ Command_Active | Status Bit | | N |
| Busy | Status Bit | | Y |
| Error | Status Bit | | Y |
| UIN3 | Status Bit | R/W head is connected | Y |
| UIN2 | Status Bit | | N |
| UIN1 | Status Bit | Data of the Tag fully read in the R/W head | Y |
| UIN0 | Status Bit | Transmitter switched on (see also WriteConfig) | Y |

The following commands exhibit divergences to the PROFIBUS specification:

### Write Config

This command is triggered in the PIB by INIT and supports Config = 1 (reset only) and Config = 3 (write data and reset). 3 bytes of config data can be written. The config data includes the possibility of switching on and off the transmitter of the read-write head. The switch on and off of the transmitter can be used in order to avoid mutual interference of R/W heads which are located closely together.

*Table 2: Configurator data*

| Byte | Bit | Meaning |
|---|---|---|
| 0 | 0..7 | Reserved, must be 0 |
| 1 | 0..7 | Reserved, must be 0 |
| 2 | 0 | 1 = Transmitter On / 0 = Transmitter Off, (default = 1) |
| | 1..7 | Reserved, must be 0 |

### Example for configuration data

“0x00, 0x00, 0x01”

### Read Config

This command reads the written Config data under Write-Config.

Possible command-specific STATUS values after faulty execution:

*Table 3: STATUS values*

| STATUS | Description of the error |
|---|---|
| DW#16#E**4**FE**82**xx | No read/write head is connected. |

### Inventory

Only parameter attributes = 0 are supported. („Inventory” page 3-29)

### Physical_Read

The parameter StartAddress and Length (+StartAddress) must be within the address range of the Tag. („Physical_Read” page 3-24)

### Physical_Write

The parameter StartAddress and Length (+StartAddress) must be within the address range of the Tag. („Physical_Write” page 3-25)

### Mem-Status

With the command Mem-Status the attribute 0x40 (physical status information) is supported. As data the response of the Tag to a GET_SYSTEM_INFORMATION command is transmitted compliant to ISO/IEC15693-3:

- Byte 0 = serial number (UID), LSB
- .
- .
- Byte 7 = serial number (UID), MSB
- Byte 8 = data carrier format (DSFID)
- Byte 9 = application identity (AFI)
- Byte 10 = memory size: Block number-1
- Byte 11 = memory size: Bytes/Block-1,
- Byte 12 = IC recognition

### Dev-Status

Only parameter attributes = 0x04 are supported. The return is a data record according to the I&M specification I&M0. The connected read/write head is described. („Dev-Status” page 3-28)

**Example:**

*Table 4: Example of the Dev-Status*

| From byte | To byte | Field | Content |
|---|---|---|---|
| 0 | 9 | Manufacturer specific header | 0, 0, 0, 0, 0,0,0,0,0,0 |
| 10 | 11 | MANUFACTURER_ID | 0x0B12 (299 = TURCK) |
| 12 | 31 | ORDER_ID | ‚TN-CK40-H1147', 0x00, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20 |
| 32 | 47 | SERIAL_NUMBER | (not supported) |
| 48 | 49 | HARDWARE _REVISION | 0x0003 |
| 50 | 53 | SOFTWARE_REVISION | ‚V' (0x56), 0x01, 0x03, 0x00 (= V1.3.0) |
| 54 | 55 | REVISION_COUNTER | (not supported) |
| 56 | 57 | PROFILE_ID | 0x5B00 (identification system, PIB profile) |
| 58 | 59 | PROFILE_SPECIFIC _TYPE | 0x0000 |
| 60 | 61 | IM_VERSION | 0x01, 0x01 (= I&M V1.1) |
| 62 | 63 | IM_SUPPORTED | 0x01, I&M0 supported |

### Next

Only parameter NextMode = 0 or 1 are supported. („Next" page 3-25)

### Get

With this command it is possible to set the write protection of a block of a Tag.

It is necessary to know the memory organisation of the Tag used (number and size of the blocks). See „Access to the data range of the data carrier" page 2-33.

**Attention**

The setting of the write protection for a block can not be undone!

*Table 5: Transmitted data to command "Get"*

| Bytes in the transmitted data field | Content |
|---|---|
| 0 | 0x02 |
| 1 | UID of the Tag, UID = 0 -> any |
| 2 | UID of the Tag |
| 3 | UID of the Tag |
| 4 | UID of the Tag |
| 5 | UID of the Tag |
| 6 | UID of the Tag |
| 7 | UID of the Tag |
| 8 | UID of the Tag |
| 9 | 0x67 |
| 10 | Block number of the blocks to be write protected (0x00 = 1st Block, 0xFF = 256th Block) |

The following data is reset if successful:

*Table 6: Received data*

| Byte in the received data field | Content |
|---|---|
| 0 | 0x02 |
| 1 | 0x67 |
| 2 | Command index |

If a fault occurs this is indicated accordingly in the STATUS.

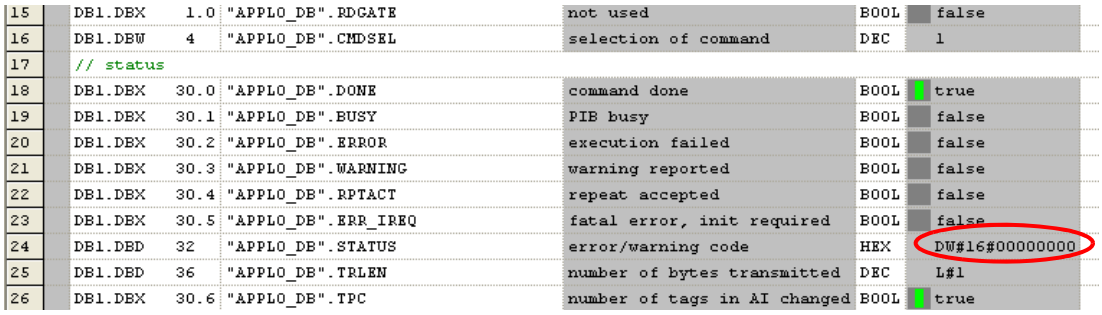
### Further commands

An overview of the further commands can be found in „3.1.3 Commands" page 3-23. Proceed just are writing and reading to execute the commands.

### Warnings and error messages

With the variables “APPLO_DB”.STATUS an error and warning code is transferred.

*Figure 23: The variable APPLO_DB. STATUS*




The following table provides information regarding the meaning of some STATUS values:

*Table 7: Status values*

| Status value of “APPLO_DB”.STATUS | Meaning of the error code |
|---|---|
| **RFID standard profile** | |
| DW#16#E**1**FE**01**xx | Tag memory error (e.g. CRC error). |
| DW#16#E**1**FE**02**xx | Dwell time of the Tag in the detection area was not long enough for successful command processing. |
| DW#16#E**1**FE**03**xx | The stated address range or command no longer matches the used Tag type. |
| DW#16#E**1**FE**04**xx | Tag is defective and must be replaced. |
| DW#16#E**1**FE**08**xx | Tag in the transfer zone does not have the expected UID. |
| DW#16#E**1**FE**09**xx | Tag does not support the current command. |
| DW#16#E**1**FE**0A**xx | At least one section of the defined range in the Tags is write protected. |
| DW#16#E**4**FE**01**xx | Supply to the read-write transceiver has been switched off due to increased current consumption, e.g. short-circuit. |
| DW#16#E**4**FE**03**xx | Antenna or transmitter of the R/W head switched off. WriteConfig necessary („Write Config” page 3-26 page 2-9, page 2-22). |
| DW#16#E**4**FE**05**xx | The requested data quantity exceeds the capacity of the internal memory. |
| DW#16#E**4**FE**06**xx | A parameter of the current command is not supported. |
| DW#16#E**4**FE**07**xx | Errors not specified in more detail have been reported by the cyclic status word (e.g. antenna out of service). The error is independent of the current command. |

Table 7: (Forts.) Status values

| Status value of “APPLO_DB”.STATUS | Meaning of the error code |
|---|---|
| ***BL ident*® specific error codes** | |
| DW#16#E**4**FE**80**xx | No R/W head is connected. |
| DW#16#E**4**FE**81**xx | The R/W head is defective |
| DW#16#E**4**FE**84**xx | Telegram content invalid (with Tag of type TW-R22-HT-B64). Range not write protected or not available. |
| DW#16#E**4**FE**88**xx | The R/W head is not supplied sufficiently |
| DW#16#E**4**FE**89**xx | The R/W head reports permanent CRC faults on the RS485 cable. EMC problem? |
| DW#16#E**4**FE**8A**xx | The Ident Unit reports permanent CRC faults on the RS485 cable. EMC problem? |
| DW#16#E**4**FE**90**xx | A command transmitted with Get is unknown to the read/write head. |
| **RFID standard profile** | |
| DW#16#E**5**FE**01**xx | The Ident Unit reports a false sequence number (SN). |
| DW#16#E**5**FE**02**xx | The PIB FB reports a false sequence number. |
| DW#16#E**5**FE**04**xx | The Ident Unit reports and invalid data block number. |
| DW#16#E**5**FE**05**xx | The PIB FB reports an invalid data block number. |
| DW#16#E**5**FE**07**xx | The PIB FB reports an invalid data block length. |
| DW#16#E**5**FE**09**xx | The Ident Unit undertakes a hardware reset (Init_Active is set to “1”), Init (Bit 15 within the cyclic “Control Word”) is expected by the PIB. |
| DW#16#E**5**FE**0A**xx | The command code “CMD” and the corresponding receive confirmation do not correspond. This is a software or synchronisation error which may normally not occur in normal operation. |
| DW#16#E**5**FE**0B**xx | The sequence of the telegrams for receive confirmation is false. |
| DW#16#E**5**FE**0C**xx | Synchronisation fault<br>(Step size of AC_H/AC_L and CC_H/CC_L in the cyclic “Control Word” is false). A new initialisation must be undertaken. |
| DW#16#E**6**FE**01**xx | Invalid command |
| DW#16#E**6**FE**02**xx | The Ident Unit reports an invalid command index. |
| DW#16#E**6**FE**05**xx | The Ident Unit reports that only write commands (Write-Config) are currently valid. |
| ***BL ident*® specific error codes** | |
| DW#16#E**6**FE**80**xx | No previous Tag recognised, no UID stored (with Next). |

*Table 7: (Forts.) Status values*

| **Status value of "APPLO_DB".STATUS** | **Meaning of the error code** |
|---|---|
| DW#16#E**6**FE**FF**xx | Unknown error |
| **RFID standard profile** | |
| DW#16#E**7**FE**01**xx | Only INIT command in this state is valid (reported by PIB). |
| DW#16#E**7**FE**02**xx | CMDSEL > CMDDIM or command code "CMD" not valid. |
| DW#16#E**7**FE**03**xx | The PIB reports:<br>Parameter "Length" of the command is too long for the global data range which is reserved within the TXBUF. |
| DW#16#E**7**FE**04**xx | RXBUF overflow (more data has been received than can be saved in the RXBUF memory). |
| DW#16#E**7**FE**05**xx | The next command must be an "INIT" command in all cases. All other commands will be rejected. |
| DW#16#E**7**FE**06**xx | The index is out of the range 111/112 and thus false. |
| DW#16#E**7**FE**07**xx | BLxx-2RFID-y does not respond to the INIT command. Check that the correct ID is set! |
| DW#16#E**7**FE**08**xx | Timeout during initialisation. |
| DW#16#E**7**FE**09**xx | A repeat of the command is not supported by the PIB*. |
| DW#16#E**7**FE**0A**xx | Fault during definition of the data package size within the PIB. |

### IEC compliant error messages

The following tables show the composition of the IEC complaint error messages. The error messages appear as a 6-digit hexadecimal value. The meaning of the error is comprised of 4 information elements which can be taken from the following three tables.

*Table 8: Positions 1 and 2 of the hex error code*

| Positions 1 and 2 of the hex error code | Meaning |
|---|---|
| 0x40 | reserved |
| 0x41 | Get_Master_Diag |
| 0x42 | Start_Seq |
| 0x43 | Download |
| 0x44 | Upload |
| 0x45 | End_Seq |
| 0x46 | Act_Para_Brct |
| 0x47 | Act_Param |
| 0x48 | Idle |
| 0x49 to 0x50 | reserved |
| 0x51 | Data_Transport |
| 0x52 to 0x55 | reserved |
| 0x56 | RM |
| 0x57 | Initiate |
| 0x58 | Abort |
| 0x59 | reserved |
| 0x5A | reserved |
| 0x5B | reserved |
| 0x5C | Alarm_Ack |
| 0x5D | reserved |
| 0x5E | Read |
| 0x5F | Write |
| 0xC0 | reserved |
| 0xC1 | FE |
| 0xC2 | NI |
| 0xC3 | AD |
| 0xC4 | EA |

*Table 8: Positions 1 and 2 of the hex error code*

| Positions 1 and 2 of the hex error code | Meaning |
|---|---|
| 0xC5 | LE |
| 0xC6 | RE |
| 0xC7 | IP |
| 0xC8 | SC |
| 0xC9 | SE |
| 0xCA | NE |
| 0xCB | DI |
| 0xCC | NC |
| 0xCD | TO |
| 0xCE | CA |
| 0xCF to 0xD0 | reserved |
| 0xD1 | Error Data_Transport |
| 0xD2 to 0xD6 | reserved |
| 0xD7 | Error Initiate |
| 0xD8 | reserved |
| 0xD9 | reserved |
| 0xDA | reserved |
| 0xDB | reserved |
| 0xDC | Error Alarm_Ack |
| 0xDD | reserved |
| 0xDE | Error Read |
| 0xDF | Error Write |

*Table 9: Positions 3 and 4 of the hex error code*

| Positions 3 and 4 of the hex error code | Meaning |
|---|---|
| 0x00 to 0x7F | reserved |
| 0x80 | DPV1 |
| 0x81 to 0xFD | reserved |
| 0xFE to 0xFF | PROFILE_SPECIFIC |

*Table 10: Positions 5 and 6 of the hex error code*

| Position 5 of the hex error code | Meaning | Position 6 of the hex error code | Meaning |
|---|---|---|---|
| **0xA**<br>10 (decimal)<br>1010 (binary) | application | 0x0 (0000) | read error |
| | | 0x1 (0001) | write error |
| | | 0x2 (0010) | module failure |
| | | 0x3 to 0x6 | not specific |
| | | 0x7 (0111) | busy |
| | | 0x8 (1000) | version conflict |
| | | 0x9 (1001) | feature not supported |
| | | 0xA to 0xF | User specific |
| **0xB**<br>11 (decimal)<br>1011 (binary) | access | 0x0 (0000) | invalid index |
| | | 0x1 (0001) | write length error |
| | | 0x2 (0010) | invalid slot |
| | | 0x3 (0011) | type conflict |
| | | 0x4 (0100) | invalid area |
| | | 0x5 (0101) | state conflict |
| | | 0x6 (0110) | access denied |
| | | 0x7 (0111) | invalid range |
| | | 0x8 (1000) | invalid parameter |
| | | 0x9 (1001) | invalid type |
| | | 0xA (1010) | backup |
| | | 0xB to 0xF | User specific |

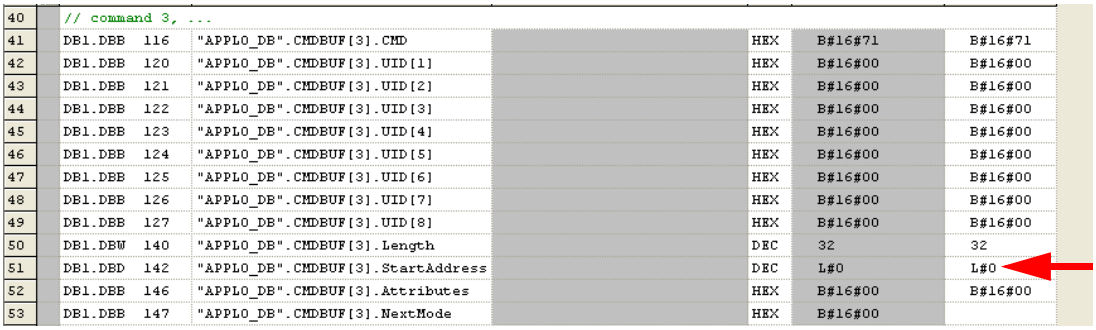
*Table 10: Positions 5 and 6 of the hex error code*

| Position 5 of the hex error code | Meaning | Position 6 of the hex error code | Meaning |
|---|---|---|---|
| **0xC**<br>12 (decimal)<br>1100 (binary) | resource | 0x0 (0000) | read constrain conflict |
| | | 0x1 (0001) | write constrain conflict |
| | | 0x2 (0010) | resource busy |
| | | 0x3 (0011) | resource unavailable |
| | | 0x4 to 0x7 | not specific |
| | | 0x8 to 0xF | user specific |
| **0xD to 0xF**<br>13 to 15 (decimal)<br>1101 to 1111 (binary) | User specific | - | - |

### Access to the data range of the data carrier

If you wish to use a data carrier other than stated in „Hardware description of the example project” page 2-2 or wish to access certain areas of the data carrier, the values “Start Address” in the „Preparing the vartable_pib0 for writing” page 2-15 or with the „Preparing the vartable_pib0 for reading” page 2-17 must be changed. The value “Start Address” is marked there with the legend D and explained.

*Figure 24: Preparing the vartable_pib0 for writing*



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 40 | | // command 3, ... | | | | | |
| 41 | | DB1.DBB 116 | "APPLO_DB".CMDBUF[3].CMD | | HEX | B#16#71 | B#16#71 |
| 42 | | DB1.DBB 120 | "APPLO_DB".CMDBUF[3].UID[1] | | HEX | B#16#00 | B#16#00 |
| 43 | | DB1.DBB 121 | "APPLO_DB".CMDBUF[3].UID[2] | | HEX | B#16#00 | B#16#00 |
| 44 | | DB1.DBB 122 | "APPLO_DB".CMDBUF[3].UID[3] | | HEX | B#16#00 | B#16#00 |
| 45 | | DB1.DBB 123 | "APPLO_DB".CMDBUF[3].UID[4] | | HEX | B#16#00 | B#16#00 |
| 46 | | DB1.DBB 124 | "APPLO_DB".CMDBUF[3].UID[5] | | HEX | B#16#00 | B#16#00 |
| 47 | | DB1.DBB 125 | "APPLO_DB".CMDBUF[3].UID[6] | | HEX | B#16#00 | B#16#00 |
| 48 | | DB1.DBB 126 | "APPLO_DB".CMDBUF[3].UID[7] | | HEX | B#16#00 | B#16#00 |
| 49 | | DB1.DBB 127 | "APPLO_DB".CMDBUF[3].UID[8] | | HEX | B#16#00 | B#16#00 |
| 50 | | DB1.DBW 140 | "APPLO_DB".CMDBUF[3].Length | | DEC | 32 | 32 |
| 51 | | DB1.DBD 142 | "APPLO_DB".CMDBUF[3].StartAddress | | DEC | L#0 | L#0 |
| 52 | | DB1.DBB 146 | "APPLO_DB".CMDBUF[3].Attributes | | HEX | B#16#00 | B#16#00 |
| 53 | | DB1.DBB 147 | "APPLO_DB".CMDBUF[3].NextMode | | HEX | B#16#00 | |

### Overview of the Turck data carriers

The data carriers of type **I-Code SL2** can be written to and read from at byte number 0 to byte number 111.

The “Start byte number” is part of the command structure „Physical_Read” page 3-24 and „Physical_Write” page 3-25 and is designated there as the “Start Address”.

The table describes the data structure of the data carrier:

- TW-I14-B128
- TW-L43-43-F-B128
- TW-L82-49-P-B128
- TW-R16-B128
- TW-R20-B128
- TW-R30-B128
- TW-R50-B128
- TW-R50-90-HT-B128
- ...

*Table 11: Data structure of I-Code SL2 data carrier*

| Byte number (Start Address) | Content | Access | Block number (a block consists of 4 bytes) |
|---|---|---|---|
| -16 to -9 | UID | Read only via „Inventory” page 3-29 | -4 to -3 |

*Table 11: Data structure of I-Code SL2 data carrier*

| Byte number (Start Address) | Content | Access | Block number (a block consists of 4 bytes) |
|---|---|---|---|
| -8 to -5 | Information about the tag | Read only via special commands | -2 |
| -4 to -1 | Conditions for write access | | -1 |
| 0 to 111 | User data range | Read / write via „Physical_Read” page 3-24 and „Physical_Write” page 3-25 | 0 to 27 |

The data carriers of type **I-Code SL1** can be written to and read from at byte number 18 to byte number 63.

The “Start byte number” is part of the command structure „Physical_Read” page 3-24 and „Physical_Write” page 3-25 and is designated there as the “Start Address”.

The table describes the data structure of the data carrier:

- TW-R16-B64
- TW-R22-HT-B64
- ...

*Table 12: Data structure of I-Code SL1 data carrier*

| Byte number (Start Address) | Content | Access | Block number (a block consists of 4 bytes) |
|---|---|---|---|
| 0 to 7 | UID | Read only via „Inventory” page 3-29 | 0 to 1 |
| 8 to 11 | Conditions for write access | Read only via special command | 2 |
| 12 to 15 | Special functions (e.g. EAS / QUIET) | Read / write via special commands | 3/4 |
| 16 | family code | | |
| 17 | application identifier | | |
| 18 to 63 | User data range | Read / write via „Physical_Read” page 3-24 and „Physical_Write” page 3-25 | 4/5 to 15 |

The data carriers of type **FRAM** can be written to and read from at byte number 0 to byte number 1999.

The “Start byte number” is part of the command structure „Physical_Read” page 3-24 and „Physical_Write” page 3-25 and is designated there as the “Start Address”.

The table describes the data structure of the data carrier:

- TW-R20-K2
- TW-R30-K2
- TW-R50-K2
- TW-R50-90-HT-K2
- ...

*Table 13: Data structure of the FRAM data carrier*

| **Byte number (Start Address)** | **Content** | **Access** | **Block number (a block consists of 4 bytes)** |
|---|---|---|---|
| 0 to 1999 | User data range | Read / write via „Physical_Read” page 3-24 and „Physical_Write” page 3-25 | 0 to 249 |
| 2000 to 2007 | UID | Read only via „Inventory” page 3-29 | 250 |
| 2008 to 2015 | AFI, DSFID, EAS | Read / write (with limitations) via special command | 251 |
| 2016 to 2047 | Special functions (e.g. EAS / QUIET) | Read only via special command | 252 to 255 |

# 3 Extract from the specification

# 1 General

TURCK publishes this excerpt of the PROFIBUS Specification „Profile for Identification Systems, Proxy Ident Function Block“ (Version 1.02, December 2005) by courtesy of the PNO (PROFIBUS Nutzer Organisation).

Please also note the „Definitions in the command and diagnostics level” page 2-21.

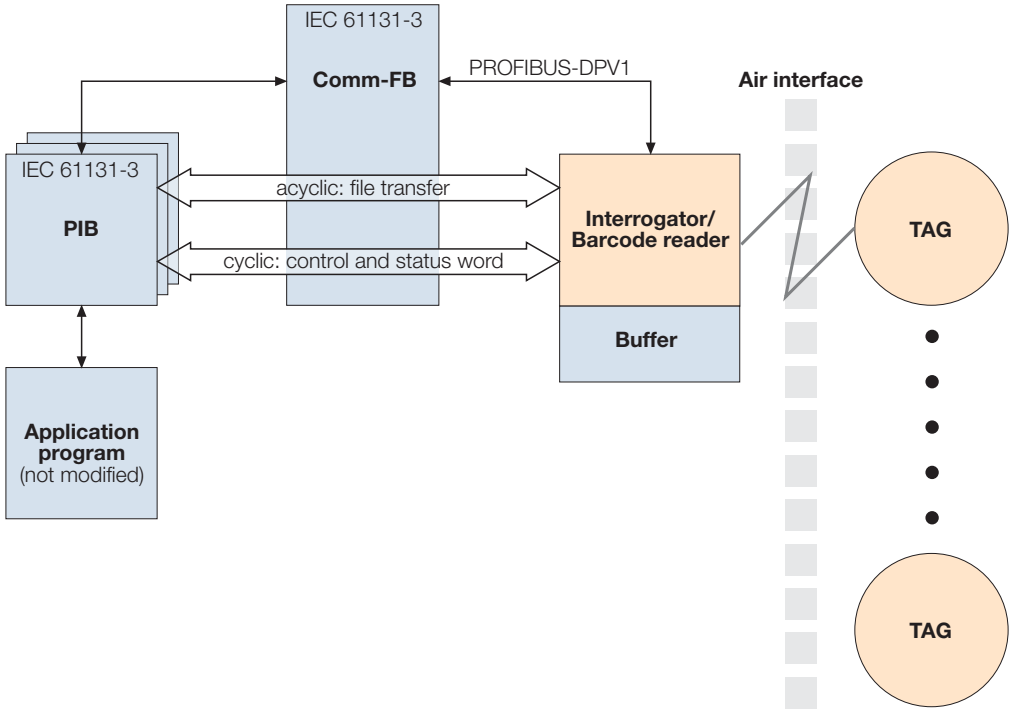
## 1.6 Function demands

In this section the function demands of the application of RFID and Barcode reader systems is defined. These demands define the limitations or limits which have to be observed with the development of the PIB and the functions which have to be realised within the field device. The RFID and barcode reader systems demands are described in parallel because of the different functions.

### 1.6.1 General demands

The basic concept involves the adaptation of existing RFID and barcode reader systems to PROFIBUS technology, so that it can be integrated into existing systems, and to make sure that use in new systems can be simplified (see following illustration).

*Figure 25: Basic concept*




As the existing proprietary solutions have to be mastered, there are certain limitations which have to be considered in addition to the PROFIBUS conformity.

The following demands must be fulfilled:

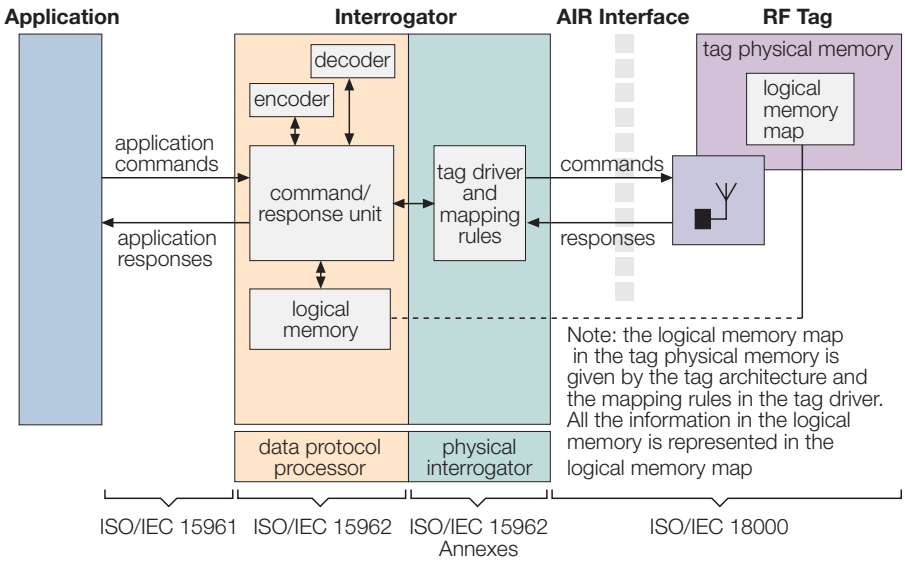
- Existing PROFIBUS systems may only be minimally affected by the integration of RFID and barcode reader systems
- The control is implemented cyclically
- The data transfer is acyclic
- Application programs should be independent of the integration of the PIB
- Fragmentation and defragmentation of data packages should occur within the PIB

### 1.6.2 Demands with the use of RFID systems

Standardization activities for RFID systems are currently underway. The objectives here include the definition of air interfaces (ISO/IEC 18000), the file format as well as handling of files (ISO/IEC 15962). See below.

This profile specification explicitly deals with data transfer via industrial networks based on the PROFIBUS as well as the integration in PLC systems.

*Figure 26: Relevant standardisation activities*

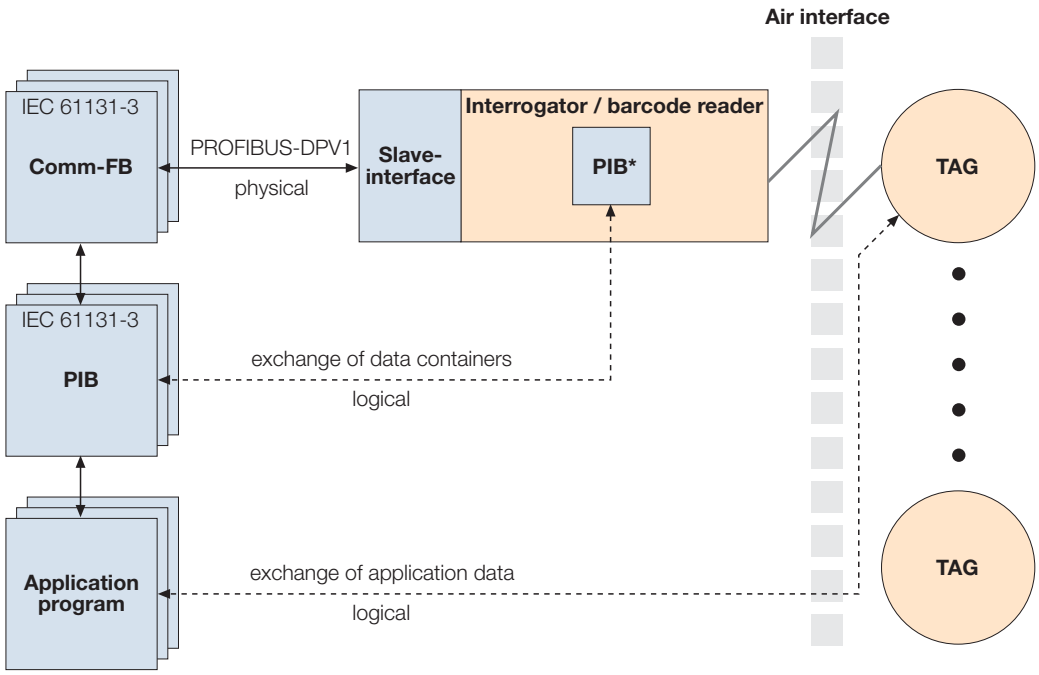



The topics concerning this specification do not need to be dealt with here as the process is ongoing. It will be possible to refer to the respective documents at a later stage when the standardisation to IEC is completed. For this reason the following points are not examined in this document:

- Air interface
- File format,
- File handler,
- Coding of application data.

At the moment the respective data is normally considered as a data package, which is interpreted both by the application in the Ident device as well as by the PLC application, which uses the PIB (see illustration below).

*Figure 27:*
*Data transfer*



## 2.1 Fundamentals of modelling

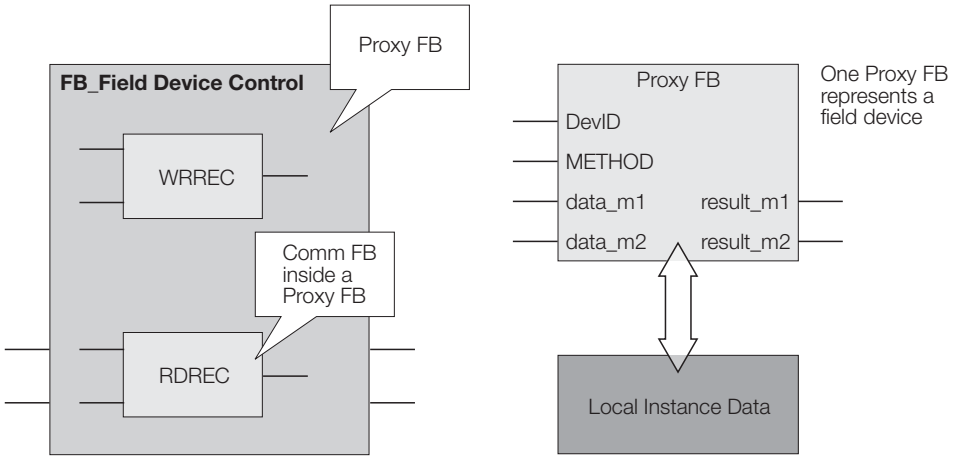
The modelling of the Proxy Ident Block (PIB) must be performed according to the following principles:

- It must correspond with the available PLC systems, e.g. by using the existing addressing concept
- It must be efficient and without overheads, i.e. the model must be performance oriented
- It must enable simple porting of the application program between different PLC systems
- It should directly use the existing Comm-FBs
- Dependencies on the hardware configuration should be avoided by good programming, i.e. such as the addressing of the application program.

## 2.2 General PIB model

The PIB is modelled as a Proxy-FB which represents a complete Ident device. Here the fundamentals of Proxy-FB modelling in accordance with [4] are observed.

*Figure 28:*
*Use of Comm-FB and Proxy-FB for PIB modelling*
*PLC: DP Master Class1*
*IEC 61131-3 program*




## 2.3 Representation

The interface for the POU type is represented by text and graphics compliant to IEC 61131-3. The behaviour of the POU is represented as a graphic state diagram with tables for the transitions and actions.

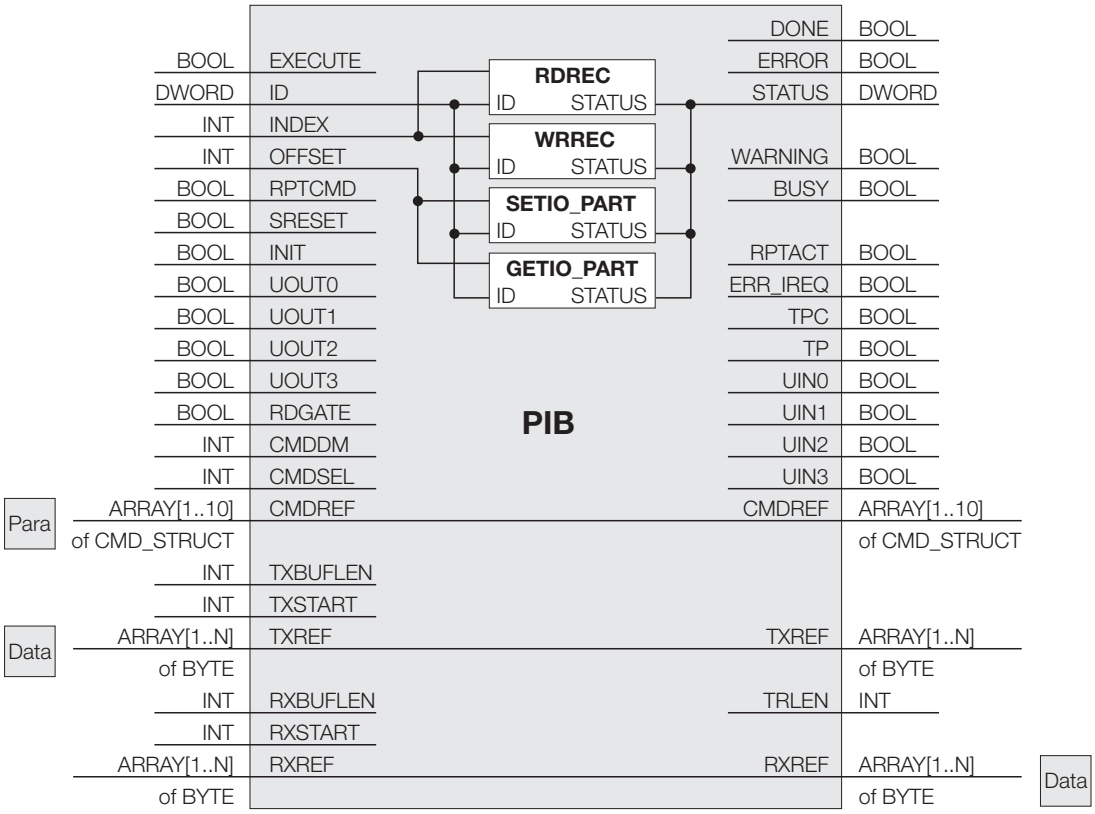
## 3 Definition of the Proxy Ident Block (PIB)

This section defines the specification of the Proxy Ident Block according to the guidelines set out in [4].

NOTE: If several PIB instances are operated simultaneously by an application program, it is necessary to ensure that the individual instances do not inhibit each other. Block definition

The following illustration is a graphic representation of the PIB interface:

*Figure 29: Graphic representation of the PIB interface*



### 3.1.1 Parameters

*Table 14:*

*PIB parameter description*

| Name | Description |
|---|---|
| EXECUTE | By setting the input parameters from the types BOOL to TRUE the user (application program) can start an FB instance. Before the FB can be put into operation the command and the corresponding parameters must be written into memory, which is assigned to the CMDREF parameter. The CMDSEL parameter must be set correctly for this purpose. This parameter is set using a positive edge. |
| ID* | The value of this input parameter is used as a unique ID for addressing of an individual device or a slot within a device. A detailed description of this is contained in document [4]. This parameter addresses an individual Ident device in conjunction with the "INDEX" parameter. |
| INDEX* | The value of this input parameter is used for identifying an individual Ident channel within a slot on a device. The use of the parameter corresponds with the definition of the address parameter "Index" according to [1] and [2]. |
| OFFSET* | The value of this input parameter is used for identifying the I/O data assigned to the channel, which is cyclically transferred as a subset of the data that is assigned to a slot (module). |
| RPTCMD* | If this parameter is set to "1", the Ident device will repeat the currently executed command or the subsequently completed command. The parameter is mirrored on the "Repeat_Command" bit in the cyclic control word (see section 4.2.1) |
| SRESET* | If this input parameter = "1", the command currently being executed in the Ident device is cancelled. The parameter is mirrored on the "Soft_Reset" in the cyclic control word (see section 4.2.1). This parameter is activated using a positive edge. |
| INIT* | With a change from "0" to "1" this input parameter forces the Ident device to start operation with the exception of the communication interface. Processes which are executed locally by the Ident device are manufacturer-specific. The parameter is mirrored on the **Init** bit in the cyclic control word (see section 4.2.1). After the sequence "Init_Active = 0 -> Init_Active = 1 Init_Active = 0" (cyclic Status word) has been completed, the PIB must send a Write-Config command to the Ident device. This parameter is set using a positive edge. |
| UOUT0* | This parameter of type BOOL represents the user-specific bit 0 which is to be transmitted within the cyclic control word (see section 4.2.1). The parameter is mirrored on bit 0 of the cyclic control word. |
| UOUT1* | This parameter of type BOOL represents the user-specific bit 1 which is to be transmitted within the cyclic control word (see section 4.2.1). The parameter is mirrored on bit 1 of the cyclic control word. |

*The application program has the task of resetting and changing all input parameters.

*Table 14: (Cont.) PIB parameter description*

| Name | Description |
|---|---|
| UOUT2* | This parameter of type BOOL represents the user-specific bit 2 which is to be transmitted within the cyclic control word (see section 4.2.1). The parameter is mirrored on bit 2 of the cyclic control word. |
| UOUT3* | This parameter of type BOOL represents the user-specific bit 3 which is to be transmitted within the cyclic control word (see section 4.2.1). The parameter is mirrored on bit 3 of the cyclic control word. |
| RDGATE* | This parameter of type BOOL represents the optional bit 8 which is to be transmitted within the cyclic control word (see section 4.2.1). The operation starts if this bit is set to "1". |
| CMDDIM* | Several commands can be saved in memory so that a more efficient application program can be written. The number of stored commands influences the memory area which is to be reserved for the respective PIB instance. The input parameterCMDDIM defines the number of "CMD_STRUCT" structures for command parameters. |
| CMDSEL* | As a certain number of parallel stored commands can exist, a means of selection must exist in order to select the individual command to be executed. The input parameter CMDSEL is used for this purpose by assigning it a predefined command. The first CMD_STRUCT is reserved for the parameter, which are assigned to the command "Write_Config". |
| CMDREF* | These In_Out parameters refer to a global memory range, which is used for storage of commands and the associated parameters. The maximum number of commands which are assigned to an individual PIB instance may not exceed 10. Section 3.1.3 describes the commands which are |
| | supported by the profile version. |
| TXBUFLEN* | These input parameters determine the number of bytes which are used by this PIB instance for storing transmit data. The number of bytes are counted by the TXSTART parameter defined position within the memory range. For reasons of consistency it is recommended that these parameters should not be changed after the installation of the PIB. |
| TXSTART* | The input parameter TXSTART defines the relative position of the "TXBUF" defined by the "TXREF" within the global memory range. This location is the start of the memory range which is assigned to the PIB instance. For reasons of consistency it is recommended that these parameters should not be changed after instantiation of the PIB. |
| TXREF* | These In_Out parameters refer to a global memory range which is used by several PIB modules. The PIB instance may share the memory with other modules. |
| RXBUFLEN* | These input parameters determine the number of bytes which are used by this PIB instance for storing receive data. The number of bytes are counted by the RXSTART parameter defined position within the memory range. For reasons of consistency it is recommended that these parameters should not be changed after instantiation of the PIB. |
| *The application program has the task of resetting and changing all input parameters. | |

Table 14: (Cont.) PIB parameter description

| Name | Description |
|---|---|
| RXSTART* | The input parameter RXSTART defines the relative position of the "RXBUF" defined by the "RXREF" within the global memory range. This location is the start of the memory range which is assigned to the PIB instance. For reasons of consistency it is recommended that these parameters should not be changed after instantiation of the PIB. |
| RXREF* | These In_Out parameters refer to a global memory range which is used by several PIB modules. The PIB instance may share the memory with other modules. |
| TRLEN | This output parameter indicates the number of the user bytes last sent (depending on the command sent or received), after the command has been successfully implemented. The number of transferred bytes from the position (TXSTART + OffsetBuffer) or (RXSTART + OffsetBuffer) is counted. |
| DONE | This output parameter is set to "TRUE" if a command with a positive result is executed. The user program can query this flag during the time that the command is being executed and before the transmitted data is computed. |
| ERROR | This output parameter is set to "TRUE" if a fault has been detected. The detected error can be locally (within the host control) or decentrally (within the Ident device). More detailed information about the error are defined in the "STATUS" parameter. This flag is set internally by the PIB in the acknowledgement telegram (Bit 0 of CI) depending on the error bit. After a renewed call of a command the flag is reset to "FALSE". |
| WARNING | This output parameter is set to "TRUE" if a warning has been detected. The detected warning can be locally (within the host control) or decentrally (within the Ident device). More detailed information about the warning is defined in the "STATUS" parameter. This flag is internally set by the PIB depending on the warn bit in the acknowledgement telegram (Bit 1..7 of CI: one or more bits are set to 1). After a renewed call of a command the flag is reset to "FALSE".<br>Note: In case of a WARNING all user data assigned to a command are transmitted correctly (if ERROR is not set). In this case the data buffer receives valid values. |
| STATUS | The output parameter "STATUS" provides detailed error information and/or warning information about the last command, which has been implemented with a negative result or a warning. The value is retained until the next time a warning or an error occurs. The coding depends on the possible source of the respective warning or error message. |
| | The following sources are possible: the embedded Comm-FBs, the functions for cyclic data access, the Ident device, the tag or the internal functions of the PIB. |
| BUSY | If this output parameter is set to "TRUE" the PIB is “busy”. A command cannot be transferred for execution (exceptions: "INIT" and "SRESET"). The parameter indicates that the PIB is still in operation. |

*The application program has the task of resetting and changing all input parameters.

*Table 14: (Cont.) PIB parameter description*

| Name | Description |
|---|---|
| RPTACT | If this output parameter is set to "TRUE", the PIB* has accepted the requirement to repeat the command currently being implemented. The cyclic input data is mirrored by the bit "Repeat_Command_Active". The output parameters remain active as long as the bit is set within the cyclic telegram. As long as the output parameter is set, the PIB* provides data after an executed command. The user is in a position to read the result of the command. |
| ERR_REQ | This output parameter is set to "TRUE" if the PIB* has detected a fatal error. It is mirrored by the Error_Flag of the cyclic input data. The output parameters remain active as long as the bit is set within the cyclic telegram. The PIB remains in the current state (state machine). If this parameter is set to "TRUE" the user must set the INIT input parameter of the PIB POU or trigger a Dev-Status command (if possible). |
| UINO | This output parameter of type BOOL represents the user-specific bit 0 which is to be transmitted within the cyclic status word (see section 4.2.2). |
| UIN1 | This output parameter of type BOOL represents the user-specific bit 1 which is to be transmitted within the cyclic status word (see section 4.2.2). |
| UIN2 | This output parameter of type BOOL represents the user-specific bit 2 which is to be transmitted within the cyclic status word (see section 4.2.2). |
| UIN3 | This output parameter of type BOOL represents the user-specific bit 3 which is to be transmitted within the cyclic status word (see section 4.2.2). |
| TP | This output parameter is set to "1" if a target is within the Ident device range. The parameter corresponds with the "Target_Present" bit of the cyclic status word (see section 4.2.2). It is reset if there is not target within the Ident device range. If an Ident device does not support this feature, the parameter is set to "0". This parameter is not used for barcode readers. |
| TPC | This output parameter is set to "1" if a new target is within the Ident device range. The parameter corresponds with the "Target_Presence_Changed" bit of the cyclic status word (see section 4.2.2). It is reset to "0" after the next "Inventory" command has been completed successfully. If an Ident device does not support this property the parameter is set to "0". This parameter is not used for barcode readers. |

*The application program has the task of resetting and changing all input parameters.

*Table 15:*

*Properties of PIB parameters*

| Name | Data type | Area | Conditions of use<br>m = mandatory, o = optional |
|---|---|---|---|
| EXECUTE | BOOL | If DONE≠O or ERROR≠O, the change of the parameter EXECUTE from "0" to "1" forces the function block to execute a command (if DONE = 0 and ERROR = 0 the PIB has not been initialized or a command is in process). | m |
| ID | DWORD | 0x00000000 .... 0xFFFFFFFF (see [4]) | m |
| INDEX | INT | 101...108, 111...118,201...254 | m |
| OFFSET | INT | 0...244 | m |
| RPTCMD | BOOL | 0 = No Repeat_Command<br>1 = Repeat_Command | o |
| SRESET | BOOL | Change from "0" to "1" forces the function block to execute the command SRESET (ends the current command). | m |
| INIT | BOOL | Change from "0" to "1" forces the function block to execute the Init procedure. | m |
| UOUT0 | BOOL | Bit 0 = 0/1 | o |
| UOUT1 | BOOL | Bit 0 = 0/1 | o |
| UOUT2 | BOOL | Bit 0 = 0/1 | o |
| UOUT3 | BOOL | Bit 0 = 0/1 | o |
| RDGATE | BOOL | 0 = Read gate not active<br>1 = Read gate active | o |
| CMDDIM | INT | 2 ... 10 | m |
| CMDSEL | INT | 1 ...10 | m |
| CMDREF | ARRAY[2.. CMDDIM] of CMD_STRUCT | CMDDIM x 42 | m |
| TXBUFLEN | INT | 0 ... 32768 | m |
| TXSTART | INT | 0 ... 32768 | m |
| TXREF | ARRAY [1..N]of BYTE | | m |
| RXBUFLEN | INT | 0 ... 32768 | m |

*Table 15: (Cont.) Properties of PIB parameters*

| Name | Data type | Area | Conditions of use<br>m = mandatory, o = optional |
|---|---|---|---|
| RXSTART | INT | 0 ... 32768 | m |
| RXREF | ARRAY [1..N] of BYTE | N | m |
| TRLEN | INT | 0 ... 32768 | m |
| DONE | BOOL | 0 = Command executed<br>1 = Command executed | m |
| ERROR | BOOL | 0 = Last command completed without error<br>1 = Last command completed with error | m |
| WARNING | BOOL | 0 = No warning information available<br>1 = Warning information available | m |
| STATUS | DWORD | See section 3.1.2 | m |
| BUSY | BOOL | 0 = PIB does not currently execute a command<br>1 = Command currently being executed by the PIB | m |
| RPTACT | BOOL | 0 = No command repetition on the PIB* active<br>1 = Command repetition on the PIB* active | o |
| ERR_REQ | BOOL | 0 = No error reported from the PIB*<br>1 = Error reported via PIB* | m |
| UINO | BOOL | Bit 0 = 0/1 | o |
| UIN1 | BOOL | Bit 0 = 0/1 | o |
| UIN2 | BOOL | Bit 0 = 0/1 | o |
| UIN3 | BOOL | Bit 0 = 0/1 | o |
| TP | BOOL | 0 = No target present<br>1 = Target present | o<br>(do not use for barcode reader) |
| TPC | BOOL | 0 = No target changed<br>1 = Target changed | o<br>(do not use for barcode reader) |

In order to simplify access to individual elements of the command structure, a common structure has been defined for each command independently of which individual parameter has been used. The used parameter depends on the definition of the respective command. These parameters are defined in section 3.1.3.

*Figure 30: Data type definitions of the PIB command:*

```
TYPE
    CMD STRUCT
        STRUCT
            CMD : BYTE;
            Config : BYTE ;
            OffsetBuffer : INT;
            UID : ARRAY[1..8] OF BYTE;
            FileName : ARRAY[1..8] OF BYTE;
            Offset : DINT;
            Length : INT;
            StartAddress : DINT;
            Attributes : BYTE;
            NextMode : BYTE;
            Timeout : INT;
            ObjectNumber : INT;
            FileType : WORD;
      END STRUCT;
END TYPE
```

### 3.1.2 Errors and warnings

The PIB reports if the required command has been executed or not executed successfully. The error message serves two purposes:

1. To change the reaction to the process, i.e. to trigger an alternative reaction, e.g. the repetition of the command at another time or at another location, or the cancellation of the process task
2. To send an alarm message to a HMI system via the user program or automatically via the PLC system.

**Note**

In the first case there are only very few alternative reactions which are dependent on the respective error message. Detailer error information is barely used.

As other function blocks (Comm-FBs) and functions are embedded in the PIB, the parameter STATUS provides status information from several sources.

The STATUS parameter can also be used for warning information.

As an alternative to the use of STATUS parameters error and warning information can be transmitted within the diagnostics data (see section 4.4).

The STATUS output of type DWORD is interpreted as a packed array consisting of four bytes as shown in the following table.

*Table 16: Structure of the STATUS output*

| Byte | Name | Definition | Data type |
|---|---|---|---|
| 0 | Function_Num | Refer to Table 17: | Byte |
| 1 | Error_Decode | Refer to Table 18: | Byte |
| 2 | Error_Code_1 | Refer to Table 19: | Byte |
| 3 | Error_Code_2 | Warnings or manufacturer-specific faults | Byte |

The definition of the byte Function_Num is based on (IEC 61158_6, Part 6; 1999) and expands the supplements realised in (PROFIBUS Communication and Proxy Function Blocks acc. to IEC 61131-3 version 1,2). It us used to group errors and warnings.

*Table 17:*

*Values from byte function Num*

| Frame Select or (Bit 7), decimal | PDU - designator (Bit 5 to 6), decimal | Error_Code_2 is used for warnings (Bit 4)* | Function_ Code / Error_Code (Bit 0 to 3) decimal | Description in accordance with this profile |
|---|---|---|---|---|
| 0 | 0 . . . 3 | 0/1 | 0 ... 15 | No error |
| 1 | 0, 1 | 0/1 | 0 . . . 15 | Error not associated with the DP-protocol and not defined for this profile |
| 1 | 2 | 0/1 | 0 . . . 15 | Error messages associated with the DP-protocol, see */2/* and [4] |
| 1 | 3 | 0/1 | 0 | Manufacturer-specific coding of Error_Code_1 and Error_Code_2 |
| 1 | 3 | 0/1 | 1 | Error_Code_1 provides error information concerning the Tag/transponder/ barcode Manufacturer-specific coding of Error_Code_2 |

*Table 17: (Cont.) Values from byte function Num*

| Frame Select or (Bit 7), decimal | PDU - designator (Bit 5 to 6), decimal | Error_Code_2 is used for warnings (Bit 4)* | Function_ Code / Error_Code (Bit 0 to 3) decimal | Description in accordance with this profile |
|---|---|---|---|---|
| 1 | 3 | 0/1 | 2 | Error_Code_1 provides error information about the air interface Manufacturer-specific coding of Error_Code_2 |
| 1 | 3 | 0/1 | 3 | Error_Code_1 provides error information about the file system Manufacturer-specific coding of Error_Code_2 |
| 1 | 3 | 0/1 | 4 | Error_Code_1 provides error information about the Ident device (Transponder/Barcode reader) Manufacturer-specific coding of Error_Code_2 |
| 1 | 3 | 0/1 | 5 | Error_Code_1 provides error information concerning communication between the PIB and Ident device (with the exception of DP errors) Manufacturer-specific coding of Error_Code_2 |
| 1 | 3 | 0/1 | 6 | Error_Code_1 provides command-specific error information Manufacturer-specific coding of Error_Code_2 |

*Table 17: (Cont.) Values from byte function Num*

| Frame Select or (Bit 7), decimal | PDU - designator (Bit 5 to 6), decimal | Error_Code_2 is used for warnings (Bit 4)* | Function_ Code / Error_Code (Bit 0 to 3) decimal | Description in accordance with this profile |
|---|---|---|---|---|
| 1 | 3 | 0/1 | 7 | Error_Code_1 provides error information which is internally generated by the PIB Manufacturer-specific coding of Error_Code_2 |
| 1 | 3 | 0/1 | 8 ... 15 | Not defined here |

* Bit 4 = 0: Error_Code_2 contains manufacturer-specific information, Bit 4 = 1: Error_Code_2 contains warning information (Byte 5 (Cl) of acknowledgement telegram)

The Error_Decode byte is used in order to define the Function_Num, Error Code 1 and Error Code 2.

*Table 18: Values of byte Error Decode*

| Error_Decode | Source | Meaning |
|---|---|---|
| 0x00 | PLC | No error, no warnings |
| 0x01 .. 0x7F | PLC | Warning (not used for this profile) |
| 0x80 | DP V1 | Errors reported compliant to IEC 61158-6 *111* |
| 0x81 .. 0x8F | PLC | 0x8x reports an error after the x-th parameter of the Comm-FB call, as defined in [4] |
| 0x90 .. 0xFD | | reserved |
| 0xFE | Profile (PIB, Ident device) | Profile-specific error |
| 0xFF | Profile (PIB, Ident device) | Reserved for future use |

Error_Code_1 provides the number which represents the error or warning. The byte Error_Decode is defined on 0xFE in the following table.

*Table 19:*

*Values of byte Error Code 1*

| Function _Code/ Error_Code* | Error_Code_1 (decimal) | Reported by | Meaning | mandatory/ optional |
|---|---|---|---|---|
| 1 | 1 | IG | Tag memory error (e.g. CRC error). | o |

*Table 19: (Cont.) Values of byte Error Code 1*

| Function _Code/ Error_Code* | Error_Code_1 (decimal) | Reported by | Meaning | mandatory/ optional |
|---|---|---|---|---|
| 1 | 2 | IG | Presence error (reported by Ident device), Tag has left the transfer window | o |
| 1 | 3 | IG | Address or command does not correspond with the Tag properties (memory size) (reported by Ident device) | o |
| 1 | 4 | IG | Tag defective, (replace Tag or battery) | o |
| 1 | 5 | IG | Tag memory overflow | o |
| 1 | 6 | IG | Unformatted Tag | o |
| 1 | 7 | IG | Inconsistent Tag data structure. Reformat Tag | o |
| 1 | 8 | IG | Tag within the transfer window does not have the expected UID (reported by Ident device) | o |
| 1 | 9 | IG | Command not supported by the Tag | o |
| 1 | 10 | IG | Access violation (e.g. element inhibited ); see ISO18000-X | o |
| 1 | 11..127 | IG | Reserved for future profile use | o |
| 1 | 128..255 | IG | Manufacturer-specific | o |
| 2 | 1 | IG | Communication timeout on the air inter-face (reported by the Ident device) | o |
| 2 | 2 | IG | More Tags/barcodes than permissible in the transfer window, (reported by Ident device) | o |
| 2 | 3..127 | IG | Reserved for future profile use | o |
| 2 | 128..255 | IG | Manufacturer-specific | o |

*Table 19: (Cont.) Values of byte Error Code 1*

| Function _Code/ Error_Code* | Error_Code_1 (decimal) | Reported by | Meaning | mandatory/ optional |
|---|---|---|---|---|
| 3 | 1 | IG | Incorrect file name (reported by Ident device) | o |
| 3 | 2 | IG | File not available (reported by Ident device) | o |
| 3 | 3 | IG | The Tag type is incorrect or not suitable for the selected operating mode. No file system available on the Tag. (reported by Ident device) | o |
| 3 | 4 | IG | Create command; no directory entries available, (reported by Ident device) | o |
| 3 | 5 | IG | Create command; file already available in directory, (reported by Ident device) | o |
| 3 | 6 | IG | Access violation (reported by Ident device) | o |
| 3 | 7 | IG | File length exceeded (reported by Ident device) | o |
| 3 | 8 | IG | File not available (falsified) (reported by Ident device) | o |
| 3 | 9..127 | IG | Reserved for future profile use | o |
| 3 | 128..255 | IG | Manufacturer-specific | o |
| 4 | 1 | IG | Voltage failure (reported by Ident device) | o |
| 4 | 2 | IG | Hardware error within the Ident device (reported by the Ident device) | o |

*Table 19: (Cont.) Values of byte Error Code 1*

| Function _Code/ Error_Code* | Error_Code_1 (decimal) | Reported by | Meaning | mandatory/ optional |
|---|---|---|---|---|
| 4 | 3 | IG | Antenna does not function (reported by the Ident device); e.g. switched off or disconnected | o |
| 4 | 4 | IG | Overflow of the command buffer of the Ident device (reported by the Ident device) | o |
| 4 | 5 | IG | Overflow of the data buffer of the Ident device (reported by the Ident device) | o |
| 4 | 6 | IG | Command not supported in this mode by the Ident device (reported by Ident device) | o |
| 4 | 7 | IG | Ident device reports an unspecific error which is reported via the cyclic status word (e.g. antenna does not function). This error is not assigned to a specific command. | o |
| 4 | 8..127 | IG | Reserved for future profile use | o |
| 4 | 128..255 | IG | Manufacturer-specific | o |
| 5 | 1 | IG | Incorrect sequence number (SN) reported by Ident device | m |
| 5 | 2 | PIB | Incorrect sequence number (SN) reported by PIB | m |
| 5 | 4 | IG | Invalid data block number (DBN) reported by Ident device | m |
| 5 | 5 | PIB | Invalid data block number (DBN) reported by PIB | m |
| 5 | 6 | IG | Invalid data block length (DBL) reported by Ident device | o |

*Table 19: (Cont.) Values of byte Error Code 1*

| Function _Code/ Error_Code* | Error_Code_1 (decimal) | Reported by | Meaning | mandatory/ optional |
|---|---|---|---|---|
| 5 | 7 | PIB | Invalid data block length (DBL) reported by PIB | m |
| 5 | 8 | IG | Command from another user is being executed (reported by Ident device) | o |
| 5 | 9 | PIB | The Ident device has implemented a hard-ware reset (InitActive set to "1"), Init (Bit 15 of the cyclic control word) expected by PIB. | m |
| 5 | 10 | PIB | The command code “CMD” and the corres-ponding acknowledge-ment do not corres-pond. This is a software or synchroni-sation error which may normally not occur in normal operation. | m |
| 5 | 11 | PIB | Incorrect sequence of the acknowledgement telegrams (TDB/DBN) | m |
| 5 | 12 | PIB | Synchronisation error (increment sizes of AC_H/AC_L and CC_H/CC_L in cyclic control word is incor-rect), INIT must be executed | m |
| 5 | 13..127 | IG | Reserved for future profile use (may not be used) | o |
| 5 | 128..255 | IG | Manufacturer-specific | o |
| 6 | 1 | IG | Invalid CMD | m |
| 6 | 2 | IG | Invalid command index CI, reported by Ident device. | m |
| 6 | 3 | IG | Invalid command para-meters (e.g. data area) reported by Ident device. | o |

*Table 19: (Cont.) Values of byte Error Code 1*

| Function _Code/ Error_Code* | Error_Code_1 (decimal) | Reported by | Meaning | mandatory/ optional |
|---|---|---|---|---|
| 6 | 4 | IG | Incorrect synchroniza-tion between user program and Tag. An expected command is missing. (Object detec-tion error) | o |
| 6 | 5 | IG | Only Write-Config is permissible in this state (reported by Ident device) | m |
| 6 | 6..127 | IG | Reserved for future profile use | o |
| 6 | 128..255 | IG | Manufacturer-specific | o |
| 7 | 1 | PIB | Only INIT command in this state is valid (reported by PIB) | m |
| 7 | 2 | PIB | Command code "CMD" is invalid | m |
| 7 | 3 | PIB | Parameter "Length" of the command is too long for the global data range which is reserved within the TXBUF. (reported by PIB) | m |
| 7 | 4 | PIB | RXBUF overflow (more received data than available memory in RXBUF) | m |
| 7 | 5 | PIB | This message informs the user that only an "INIT" is permissible as the next command. All other commands are rejected. | m |
| 7 | 6 | PIB | Incorrect index (out of range: 101 .. 108) | m |
| 7 | 7 | PIB | Ident device does not respond to INIT (Init_Active in cyclic status telegram expected) | m |

*Table 19: (Cont.) Values of byte Error Code 1*

| Function _Code/ Error_Code* | Error_Code_1 (decimal) | Reported by | Meaning | mandatory/ optional |
|---|---|---|---|---|
| 7 | 8 | PIB | Timeout during the Init process (60 seconds as defined in TC3WG9) | m |
| 7 | 9 | PIB | Command repetition not supported by PIB* | o |
| 7 | 9..127 | PIB | Reserved for future profile use | o |
| 7 | 128..255 | PIB | Manufacturer-specific | o |

*) Bit 0 to 3 (decimal coded) of Function_Num IG ... Ident device

The byte Error_Code_2 provides either warning information (if bit 4 of Function_Num is set to "1") or optionally a manufacturer-specific error number (if bit 4 of function_Num is reset to "0"). The warnings are mirrored by byte 5 (CI) of the acyclic acknowledgement telegram.

*Table 20: Coding of warnings*

| Error_Decode | Error_Code_2 (Bit position) | Source | Meaning |
|---|---|---|---|
| 0xFE | 0 | Ident device | Bit fixed setting to "0" |
| 0xFE | 1 | Ident device | Manufacturer-specific |
| 0xFE | 2 | Ident device | If the Ident device is a barcode reader: manufacturer-specific<br>If the Ident device is an RFID device: Battery almost empty |
| 0xFE | 3 | Ident device | Manufacturer-specific |
| 0xFE | 4 | Ident device | Manufacturer-specific |
| 0xFE | 5 | Ident device | Manufacturer-specific |
| 0xFE | 6 | Ident device | Manufacturer-specific |
| 0xFE | 7 | Ident device | Manufacturer-specific |

### 3.1.3 Commands

In this section commands are described which are supported by the PIB POU and their respective parameters. The following limitations apply for the use of commands:

- The cyclic control process has precedence over acyclically transmitted commands
- "INIT" and "SRESET" interrupt the execution of a command within the Ident device
- After transmission of a cyclic control telegram (INIT, SRESET) the continuous change of the "DONE" parameter relates to the cyclic control telegram and not to the INIT or SRESET interrupted command
- "INIT" resets the communication (cyclic control flow and status flow, cyclic commands) between the PIB and the Ident device. For this procedure the "Init" is first of all transmitted within the cyclic control word. After reset of the "Init_Active" bit in the status word a "Write-Config" command is triggered and sent to the Ident device. For this the user must provide "Write-Config" parameters within the command range before the "INIT" command is requested. (see section 4.2.3)
- The "Write-Config" command resets all functions in the Ident device with the exception of the communication
- The "SRESET" command ends the last command

*Table 21:*

*Description of the PIB commands*

| **Name** | **Description** | |
|---|---|---|
| Get | Manufacturer-specific data are read into the Ident device with this command. Hereby the TXBUF range is used as a manufacturer-specific memory range for the parameter data (optional transmission data). Received data are stored in the RXBUF area from the start of the range. The parameter TRLEN of the PIB POU indicates the number of received bytes. | |
| | VAR<br>CMD : BYTE := 0x62; (* b *)<br>END_VAR | |
| | **Parameters** | **Description** |
| | OffsetBuffer | This parameter defines the relative offset in the TXBUF. Herewith the address is defined in the memory range, in which the first byte of parameter data to be sent is stored. Further parameter data is stored accordingly in a consistent manner. |
| | Length | This parameter defines the number of bytes to be sent to the Ident device, which commence with the parameter defined by the OffsetBuffer. This range is between the following values: 0 ... 226. |
| Physical_Read | The data is read from a transponder which are defined via a physical start address and the length of the data to be read with this command. The parameter TRLEN of the PIB POU indicates the number of received bytes. | |
| | VAR<br>CMD : BYTE := 0x70; (* p *)<br>END_VAR | |
| | **Parameters** | **Description** |
| | OffsetBuffer | This parameter defines the relative offset in the RXBUF. Herewith the address is defined in the memory range, in which the first byte of parameter data to be received is stored. All the following bytes must be stored at incremented addresses. |
| | UID | This parameter identifies an individual transponder.<br>UID = 0: Any (no specific transponder) The currently available Tag is read. |
| | Length | This parameter defines the number of bytes to be read from the transponder which begin with the address defined by the StartAddress parameter. |
| | Start Address | This parameter defines a physical address within a transponder memory. |

*Table 21: (Cont.) Description of the PIB commands*

| Name | Description | |
|---|---|---|
| Physical_Write | The data is written to a transponder which is defined via a physical start address and the length of the data to be written with this command. | |
| | VAR<br>CMD : BYTE := 0x71; (* q *)<br>END VAR | |
| | **Parameters** | **Description** |
| | OffsetBuffer | This parameter defines the relative offset in the TXBUF. Herewith the address is defined in the memory range, in which the first byte of parameter data to be sent is stored.. |
| | UID | This parameter identifies an individual transponder.<br>UID = 0:<br>Any (no specific transponder)<br>The currently available Tag is read. |
| | Length | This parameter defines the number of bytes to be sent to the transponder which begin with the address defined by the StartAddress parameter. |
| | Start Address | This parameter defines a physical address within a transponder memory. |
| Next | With this command the procedures are ended with a transponder. The Next command is executed when the next transponder is detected/signalled. | |
| | VAR<br>CMD : BYTE := 0x6E; (* n *)<br>END VAR | |
| | **Parameters** | **Description** |
| | UID | This parameter identifies an individual transponder.<br>UID = 0:<br>Any<br>The currently available Tag is read. |
| | NextMode | Valid values:<br>NextMode = 0 (the Next command applies for all (another or the same) Tags)<br>NextMode = 1 (only another Tag is dealt with) |

*Table 21: (Cont.) Description of the PIB commands*

| Name | Description | |
|---|---|---|
| Write Config | The operation of the Ident device is modified with this service. The communication will continue. Herewith new parameters can be sent to the Ident device (ConfigData). A reset can also be triggered with it where the Ident device is requested to restart operation. TXBUF is used as the manufacturer-specific area for configuration data, as configuration data (Config data) are manufacturer-specific.<br>Normally the Write-Config command is automatically implemented by the PIB during the INIT phase. The Write Config command can be started optionally with EXECUTE.<br>Config is forcibly supported:<br>Config = 1<br>Config is optionally supported:<br>Config = 2 or Config = 3 | |
| | VAR<br>CMD : BYTE : = 0x78; (* x *)<br>END VAR | |
| | **Parameters** | **Description** |
| | OffsetBuffer | This parameter defines the relative offset in the TXBUF. Herewith the address is defined in the memory range, in which the first byte of data to be sent is stored. |
| | Length | With this parameter the number of "config data" bytes written to the Ident device is determined. |
| | Config | Config = 0...not permitted<br>Config = 1...Reset, no ConfigData<br>Config = 2...No reset, ConfigData are sent<br>Config = 3 Reset, ConfigData is sent<br>Config > 3 ... reserved |
| | Definition of the sub-parameters provided in the response: | |
| | MaxPacketSize | This parameter is sent from the PIB* to the PIB. It contains information concerning the maximum length of the Ident PDU (Ident header + data), which the slave can receive or transmit. Based on this parameter, the PIB dynamically determines the PDU length supported by the PDU* during the initialisation phase (INIT) and sends it to the internal algorithm for packaging data and sets it according to the PDU length.<br>00 = Standard (corresponds to 240 bytes) 64. ..240 = max. permissible PDU length within the PIB*<br>01... 63 = reserved<br>241... 255 = reserved |

*Table 21: (Cont.) Description of the PIB commands*

| Name | Description | |
|---|---|---|
| Read Config | With this service the configuration data are read from the Ident device. RXBUF is used as the manufacturer-specific area for configuration data, as configuration data (Config data) are manufacturer-specific. The parameter TRLEN of the PIB POU indicates the number of received bytes. | |
| | VAR<br>CMD : BYTE := 0x61; (* a *)<br>END VAR | |
| | **Parameters** | **Description** |
| | OffsetBuffer | This parameter defines the relative offset in the RXBUF. Herewith the address is defined in the memory range, in which the first byte of data to be read is stored. |
| Mem-Status | With this service the status of a Tag (battery state, memory size, available capacity) are read. The RXBUF is used as the manufacturer-specific area for status data, as the status data is manufacturer-specific. The parameter TRLEN of the PIB indicates the number of received bytes. | |
| | VAR<br>CMD : BYTE := 0x73; (* s *)<br>END VAR | |
| | **Parameters** | **Description** |
| | UID | This parameter identifies an individual transponder.<br>UID = 0:<br>Any (no specific transponder).<br>The currently available Tag is read. |
| | Attributes | This parameter defines the class of the information to be read. The following values are valid:<br>0x00...reserved<br>0x01...Warn Info<br>0x02...reserved<br>0x03...reserved<br>0x04...physical status information (manufacturer specific detailed information)<br>0x05...Status information for file system (manufacturer specific detail information)<br>0x06 - 0x7F reserved<br>0x80 - 0xFF manufacturer-specific |
| | OffsetBuffer | This parameter defines the relative offset in the RXBUF. Herewith the address is defined in the memory range, in which the first byte of data to be read is stored. |

*Table 21: (Cont.) Description of the PIB commands*

| **Name** | **Description** | |
|---|---|---|
| Dev-Status | The status of an Ident device is read with this service. The RXBUF is used for status data as the manufacturer-specific area, as the status data is manufacturer-specific. The parameter TRLEN of the PIB POU indicates the number of received bytes. | |
| | VAR<br>CMD : BYTE : = 0x74; (* t *)<br>END VAR | |
| | **Parameters** | **Description** |
| | Attributes | This parameter defines the class of the information to be read. The following values are valid:<br>0x00...reserved<br>0x01... Warn Info (manufacturer-specific detailed information)<br>0x02... Error history (manufacturer-specific detailed information)<br>0x03... Command history (manufacturer-specific detailed information)<br>0x04... channel-related I&M Info (data record I&M0)<br>0x05... channel-related I&M Info (data record I&M1)<br>0x06... channel-related I&M Info (data record I&M2)<br>0x07... channel-related I&M Info (data record I&M3)<br>0x08... channel-related I&M Info (data record I&M4)<br>0x09 - 0x7F reserved<br>0x80 - 0xFF manufacturer-specific |
| | OffsetBuffer | This parameter defines the relative offset in the RXBUF. Herewith the address is defined in the memory range, in which the first byte of data to be read is stored. |

*Table 21: (Cont.) Description of the PIB commands*

| Name | Description |
|---|---|
| Inventory | This command is used in order to request a list of all UIDs which can be contacted within the antenna beam. The possibility for manufacturer-specific additional information is provided. The RXBUF has the following structure.<br><br>The following example indicates the structure of the file to be sent and should not be used as structured text in the PLC program. In the example 5 objects (ObjectNumber = 5) with ObjectLength = 16 are transmitted.<br>VAR CONSTANT<br>ObjectNumber : INT : = 5;<br>ObjectLength : INT : = 16;<br>END_VAR<br><br>TYPE<br>UID_STRUCT<br>STRUCT<br>UID : ARRAY[1..8] OF BYTE;<br>Data : ARRAY[1.. (ObjectLength-8) ] OF<br>BYTE;<br>END STRUCT;<br>END_TYPE<br><br>TYPE<br>UidList: ARRAY[1..ObjectNumber] OF<br>UID _STRUCT;<br>END_TYPE |
| | VAR<br>CMD : BYTE : = 0x69; (* i *)<br>END VAR |

| Parameters | Description |
|---|---|
| Attributes | This parameter defines the class of the information to be read. The following values are valid:<br>0x00...All UIDs are read (without additional information)<br>0x01 - 0x7F reserved<br>0x80 - 0xFF manufacturer-specific |
| OffsetBuffer | This parameter defines the relative offset in the RXBUF. Herewith the address is defined in the memory range, in which the first byte of data to be read is stored. |

*Table 21: (Cont.) Description of the PIB commands*

| Name | Description | |
|---|---|---|
| | **Definition of the sub-parameters provided in the response:** | |
| | **Parameters** | **Description** |
| | Object Number | This parameter defines the number of UIDs which are provided in the acknowledgement telegram. |
| | Object Length | This parameter defines the number of bytes which are assigned in an individual UID (UID length + additional data).<br>With attributes =0x00 the following applies: ObjectLength = 8. |
| | UidList: | These optional parameters contain a list of manufacturer-specific information which is assigned to the UIDs which are currently accessible within the antenna beam. |

*Table 22: Overview*

| Octet | Coding / Data | Meaning | Notes / Remedy |
|---|---|---|---|
| n+4 | Bit 7...0: Channel number | Channel number within the module to which the Status_Message relates. | |
| n+5 | Bit 7...0: Function_Num | See table 11 for the definition of the coding. | Expansion of the Ident-specific profile |
| n+6 | Bit 7...0: Error_Decode | See table 12 for the definition of the coding. | Expansion of the Ident-specific profile |
| n+7 | Bit 7...0: Error_Code_1 | See table 13 for the definition of the coding. | Expansion of the Ident-specific profile |
| | Bit 7...0: Error_Code_2 | See table 13 for the definition of the coding. | Expansion of the Ident-specific profile |

### 3.1.6. Timing of the PIB

The following diagram shows the control timing of the PIB:

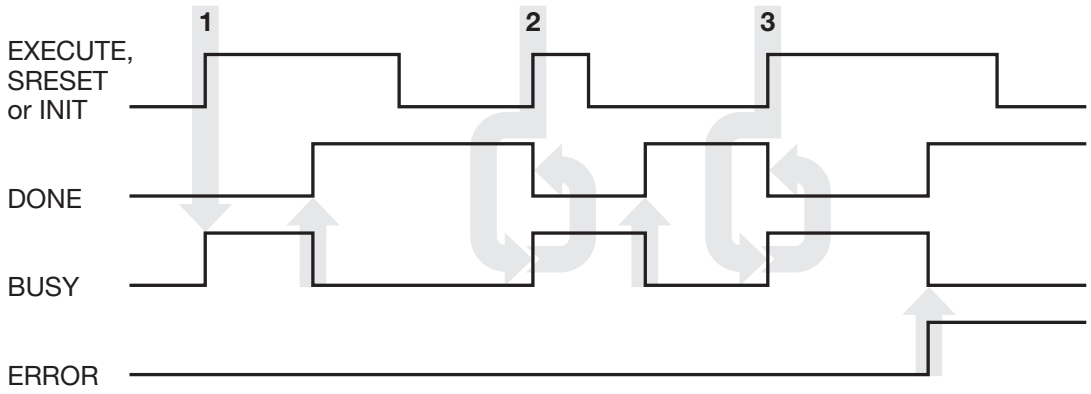



*Figure 31: General control timing of the PIB*

**1** The variable EXECUTE/INIT/SRESET remains “1” for the entire time, which the respective function block requires for processing of the command. Only after DONE = 1 has occurred, the command is reset by the user to “0”.

**2** The command EXECUTE/INIT/SRESET is only triggered here by a short pulse. The execution of the command is **not** aborted.

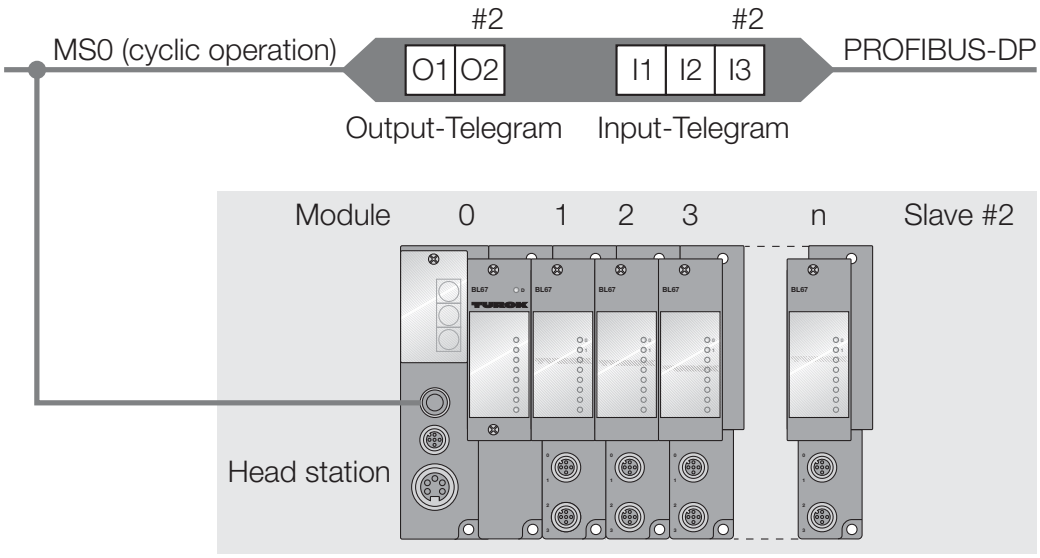
**3** As for 1), however the execution of the command is aborted because an error has occurred.

## 4.5 Data access within field device

### 4.5.1 General device model

A PROFIBUS device is either laid-out as a compact or a modular device (see figure below). In case of a modular device it is constructed of a head station and a number of modules. It may optionally have binary inputs and outputs. The granularity of devices and modules may vary.

*Figure 32: General device model*




**Head station**

The head station contains the interface to PROFIBUS and the corresponding parameters. The head station can be redundant. In this case, the redundant head station may occupy a backplane slot different from 0.

**Module**

A module usually contains a logical or physical combination of channels. The modules are inserted into the slots, or are otherwise stacked. A module may contain several channels. Modules are counted continuously, starting with the module next to the head station. There are modules, without channels (e.g. power supply). A module is logically identified by a slot number as shown in the figure above. The slot number is used for addressing purpose.

**Channel**

A channel contains all hardware and software components used to produce a measurement value (input channel) or to create a physical output signal (output channel). Thus, the ident channels represent the Ident Unit. Channels are counted separately for each module, starting with the first channel 1. A module may contain up to 8 ident channels.

### 4.5.2 Block mapping for cyclic PROFIBUS-DP data transfer

The output values from master to the slave and the input values from the connected sensors are transferred as cyclic data. Cyclic data will be transferred via the Data_Exchange service ("...cyclic exchange of the I/O data with the DP-Master (Class 1)..."). The input and output values can be mapped into the cyclic data stream. This mapping is device-global.

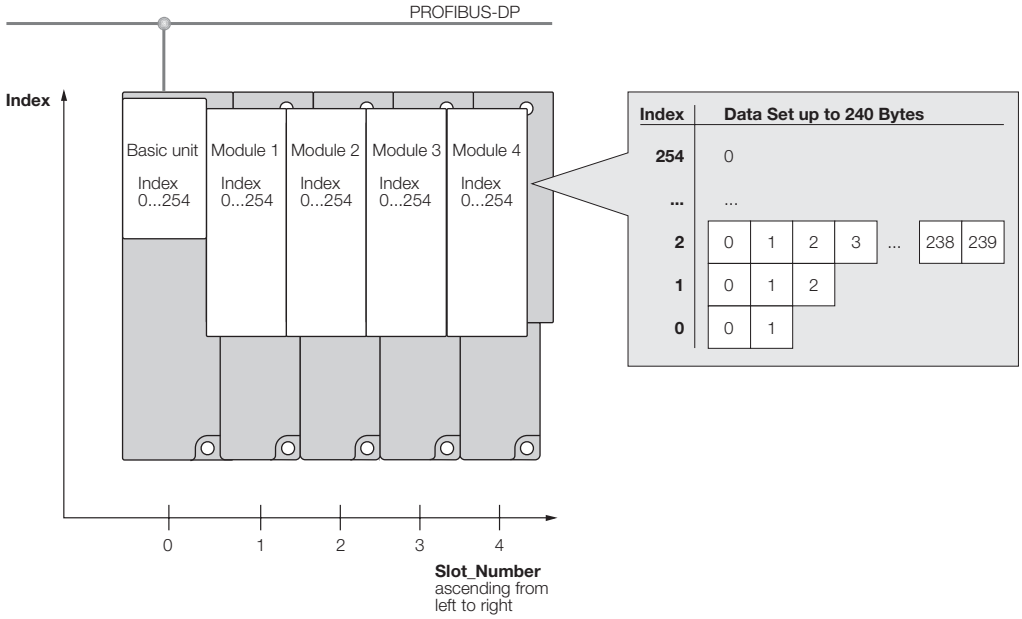
The mapping relies on the modular structure of the GSD [7]. The modules in a device are defined using the Module/End Module keywords. The assignment to a slot is controlled by the values inside the SlotDefinition/EndSlotDefinition keywords. The assignment of modules to physical slots is done by the configuration device.

### 4.5.3 Block mapping for acyclic PROFIBUS-DP data transfer

The acyclic PROFIBUS read and write services are based on slot index address combinations. Within this profile a mapping is defined between the device structure and the slots. The device block is always assigned to slot number 0 (zero). Each module of a device is mapped to one slot.

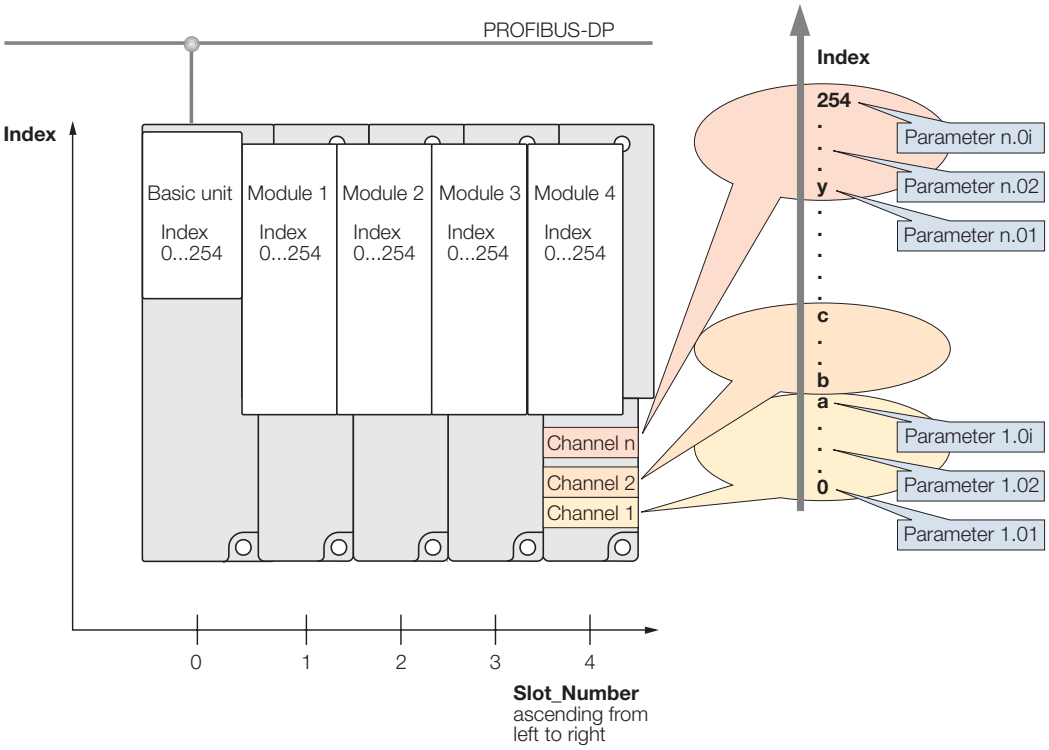
Each parameter of a module has to be mapped to an index in order to be accessible by MS1/ MS2 services.

*Figure 33: Mapping of input and output values to cyclic communication*



In case several channels are assigned to one module, they will share the addressing space of the related slot. All parameters, independent from the channel they are assigned to, have to become addressed by an individual index as illustrated below..

*Figure 34: Assignment of parameters and addresses*




### 4.5.4 Ident channel definition

This chapter describes the mapping of ident channels to modules. Basically it is possible that up to 8 ident channels can be assigned to an individual module. Module 0 is not used to support an ident channel. For each of the ident channels two indices are used for acyclic communication.

Indices 101 to 108 are used to parameterize one ident channel. Only the Write_Config command is allowed to be transferred through these indices. After the module start up, a Write-Config must be sent to each ident channel. Only after implementing this command will the channel be ready for operation. Even during normal operation a Write-Config is accepted. Running commands are interrupted by Write-Config. The user receives no further acknowledgment for the interrupted command.

- Index 11x ("x" representing the channel number)

Indices 111 to 118 are used for sending the actual commands and related acknowledgments (all commands except Write_Config).

The indices of a module are used as follows:

*Table 23: Indices of a module*

| Index | Usage |
|---|---|
| 0 ... 100 | reserved |
| 101 | Parameterisation channel 1 (CHL = 1) |
| 102 | Parameterisation channel 2 (CHL = 2) |
| 103 | Parameterisation channel 3 (CHL = 3) |
| 104 | Parameterisation channel 4 (CHL = 4) |
| 105 | Parameterisation channel 5 (CHL = 5) |
| 106 | Parameterisation channel 6 (CHL = 6) |
| 107 | Parameterisation channel 7 (CHL = 7) |
| 108 | Parameterisation channel 8 (CHL = 8) |
| 109, 110 | reserved |
| 111 | Data transfer channel 1 (CHL = 1) |
| 112 | Data transfer channel 2 (CHL = 2) |
| 113 | Data transfer channel 3 (CHL = 3) |
| 114 | Data transfer channel 4 (CHL = 4) |
| 115 | Data transfer channel 5 (CHL = 5) |
| 116 | Data transfer channel 6 (CHL = 6) |
| 117 | Data transfer channel 7 (CHL = 7) |
| 118 | Data transfer channel 8 (CHL = 8) |
| 119 ... 200 | reserved |
| 201 ... 254 | manufacturer-specific |
| 255 | I&M functions |

## 5 Identification & maintenance functions (I&M functions)

The main purpose of the I&M functions is to support the end user during various scenarios of a device's life cycle be it configuration, commissioning, parameterization, diagnosis, repair, firmware update, asset management, audit trailing, and alike. Well-defined uniform parameters and rules should enable the manufacturers to offer devices that behave in a uniform manner and that should enable the end user to act quick and directly. Information provided by the I&M function may be related to the complete device, to an individual module within a device are a channel associated to a module. Data structures and access mechanisms for device and module related I&M functions are defined in "PROFIBUS Profile Guideline, Part 1, Identification and Maintenance Functions" [5].

Addressing of channel related I&M functions is currently under discussion within TC3 of the PNO. The next version of the "PROFIBUS Profile Guideline, Part 1, Identification and Maintenance Functions" will define how to access these information. As it is the intention of the work group WG9 TC3 not to define a proprietary version of channel related I&M functions this new version of "PROFIBUS Profile Guideline, Part 1, Identification and Maintenance Functions" shall become applicable here.

### 5.1 PROFILE_ID

The Profile ID's used for the profile "PROFIBUS Proxy Ident Function Blocks according to IEC 61131-3" are:

- 5B00H for RFID systems and
- 5B10H for barcode readers.

Assignment of Profile ID's is managed by PNO.

### 5.2 Channel related information

For each channel it is possible to provide channel related I&M functions. Thus, a device may contain as many channel related information, as channels may be implemented in the individual modules. he scope of this information is restricted to an individual channel. The semantic meaning of the channel related I&M information is the same as of those related to a complete device or module. They are represented by the data types described in [5].

## Annex A – Conformity table

The following table contains a list of the permissible "Implementation independent properties" of the PIB. A manufacturer who declares conformity with this PNO specification, must provide a list of all compliant properties of the supported PIB in this tabular format.

*Table 24: Implementation independent properties for the PIB*

| Number | Property | Selected implementation |
|---|---|---|
| 3 | Maximum supported memory size for the TXBUF | |
| 3 | Maximum supported memory size for the RXBUF | |
| 3 | Maximum number of commands which should be stored in the CMDREF | |
| | | |
| | | |
| | | |

The following tables contain a listing of the communication relevant properties of the Ident device, which are relevant for this profile specification. The manufacturer of an Ident device, which communicates via a PIB instance, must provide a list of all compliant properties of the supported Ident device in this tabular format.

*Table 25: Conformity table for Ident devices*

| Number | Property | Explanation | Implementation-specific additional information | Conform? (Y/N) |
|---|---|---|---|---|
| 3.1.3 | Read | Command | | |
| 3.1.3 | Get | Command | | |
| 3.1.3 | Physical_Read | Command | | |
| 3.1.3 | Write | Command | | |
| 3.1.3 | Put | Command | | |
| 3.1.3 | Physical_Write | Command | | |
| 3.1.3 | Format | Command | | |
| 3.1.3 | Create | Command | | |
| 3.1.3 | Delete | Command | | |
| 3.1.3 | Clear | Command | | |
| 3.1.3 | Update | Command | | |
| 3.1.3 | Next | Command | | |
| 3.1.3 | Get-Directory | Command | | |

*Table 25: (Cont.) Conformity table for Ident devices*

| Number | Property | Explanation | Implementation-specific additional information | Conform? (Y/N) |
|---|---|---|---|---|
| 3.1.3 | Set-Attribute | Command | | |
| 3.1.3 | Get-Attribute | Command | | |
| 3.1.3 | Write Config | Command | | |
| 3.1.3 | Read Config | Command | | |
| 3.1.3 | Mem-Status | Command | | |
| 3.1.3 | Dev-Status | Command | | |
| 3.1.3 | Inventory | Command | | |
| 3.1.3 | Read-BarCode | Command | | |
| 4.2.1 | Reading_Gate | Control Bit | | |
| 4.2.1 | Command repetition | Control Bit | | |
| 4.2.2 | Target_Presence_Changed | Status Bit | | |
| 4.2.2 | Target_Present | Status Bit | | |
| | MaxPacketSize | Max. supported size of the Ident PDU | | |

## Annex B - Elementary data types of this specification

This annex contains information about defined data types which are used in the profile specification.

*Table 26:*

*Data types*

| Name | Definition | Reference source |
|---|---|---|
| BOOL | Boolean (the possible values for variables of this data type must be 0 and 1, corresponding the keywords FALSE and TRUE.) | IEC 61131-3 |
| DWORD | Bit sequence of length 32 | IEC 61131-3 |
| WORD | Bit sequence of length 16 | IEC 61131-3 |
| INT | Interger (the value range for variables of this data type ranges from - ($2^{16-1}$) bis ($2^{16-1}$)-1.) | IEC 61131-3 |
| ANY | | IEC 61131-3 |
| DINT | Double integrer (the value range for variables of this data type ranges from - ($2^{32-1}$) to ($2^{32-1}$)-1.) | IEC 61131-3 |
| BYTE | Bit sequence of length 8 | IEC 61131-3 |
| ARRAY[1..x] of Data Type | | IEC 61131-3 |
| CMD_ STRUCT | TYPE<br>CMD STRUCT<br>STRUCT<br>CMD : BYTE;<br>Config : BYTE;<br>OffsetBuffer : INT;<br>UID : ARRAY[1..8] OF BYTE;<br>FileName : ARRAY[1..8] OF BYTE;<br>Offset : DINT;<br>Length : INT;<br>StartAddress : DINT;<br>Attributes : BYTE;<br>NextMode : BYTE;<br>Timeout : INT;<br>ObjectNumber : INT;<br>FileType : Word;<br>END STRUCT;<br>END TYPE | |
| DIRELEMENTS_STR UCT | TYPE<br>DIRELEMENTS_STRUCT<br>STRUCT<br>FileName : ARRAY[1..8] OF BYTE;<br>UsedLength : DINT;<br>Attributes : BYTE;<br>FileLength : DINT;<br>FileType : WORD<br>END STRUCT;<br>END_TYPE | |

*Table 26: (Cont.) Data types*

| Name | Definition | Reference source |
|---|---|---|
| DIRLIST_ STRUCT | Even though this structure is defined in the ST language it should not be copied into an ST program, as it contains an array (field) with a dynamic length, which is not IEC 61131-3 (ST) compliant. ST has only been selected in the document for reasons of consistency.<br><br>TYPE<br>DIRLIST_ STRUCT<br>STRUCT<br>UID1 : ARRAY[1..8] OF BYTE;<br>TagName : ARRAY[1..8] OF BYTE;<br>FreeUserMem : DINT;<br>Checksum : WORD<br>FileCount : INT;<br>FileList : ARRAY[1..FileCount] of<br>DIRELEMENTS_STRUCT;<br>END STRUCT;<br>END TYPE | |
| UID_STRUCT | Even though this structure is defined in the ST language it should not be copied into an ST program, as it contains an array (field) with a dynamic length, which is not IEC 61131-3 (ST) compliant. ST has only been selected in the document for reasons of consistency.<br><br>TYPE<br>ObjectLength : INT;<br>END_TYPE<br>TYPE<br>UID STRUCT<br>STRUCT<br>UID : ARRAY[1..8] OF BYTE;<br>Data : ARRAY[1..(ObjectLength-8)] OF<br>BYTE;<br>END_STRUCT;<br>END_TYPE | |
| UidList: | Even though this structure is defined in the ST language it should not be copied into an ST program, as it contains an array (field) with a dynamic length, which is not IEC 61131-3 (ST) compliant. ST has only been selected in the document for reasons of consistency.<br><br>TYPE<br>ObjectNumber : INT;<br>END TYPE<br><br>TYPE<br>UldList: ARRAY[1..ObjectNumber] OF<br>UID_STRUCT;<br>END TYPE | |

The coding of a UID is defined as follows in the Technical Report ISO/IEC /TR 15963, Automatic identification - Radio Frequency Identification for item management - Unique identification for RF tags, Annex A:

*Table 27: Coding of a UID*

| AC (Allocation Class) | UID issuer Registration Number | Serial number |
|---|---|---|
| 8 bit | Size of AC_value defined | Size of AC_value defined |
| MSB | | LSB |

*Table 28: AC values*

| AC value | Class | UID issuer identifier size | Size of the serial no. | Registration body (the "UID issuer Registration Number") |
|---|---|---|---|---|
| '11100000' | 7816-6 | 8 bit | 48 bit | APACS (ISO/IEC 7816-6 registration body) |
| '11100001' | 14816 | perNEN | perNEN | NEN (ISO 14816 registration body) |
| '11100010' | EAN.UCC | per EAN.UCC | per EAN.UCC | EAN.UCC |
| 000xxxxx | INCITS 256 | per ANS INCITS 256 | per ANS INCITS 256 | ANSI ASC INCITS T6 |
| '11100011' to '11101111' | RFU | not applicable | not applicable | Reserved for future ISO use |

# 4 Glossary

A

**Acknowledge**

Acknowledgment of a signal received..

**Active metal component**

Conductor or conducting component that is electrically live during operation.

**Address**

Identification number of, e.g. a memory position, a system or a module within a network.

**Addressing**

Allocation or setting of an address, e.g. for a module in a network.

**Analogue**

Infinitely variable value, e.g. a voltage – which operates proportionally. The value of an analogue signal can take on any value, within certain limits.

**ARP**

Used to definitely allocate the hardware addresses (MAC-IDs) assigned worldwide to the IP addresses of the network clients via internal tables.

**Automation device**

A device connected to a technical process with inputs and outputs for control. Programmable logic controllers (PLC) are a special group of automation devices.

B

**Baud**

Baud is a measure for the transmission speed of data. 1 Baud corresponds to the transmission of one bit per second (bit/s). If one bit is transmitted per step, the baud rate is identical with the transmission speed in bits/second.

**Baud rate**

See “Baud”.

**Bidirectional**

Working in both directions.

**Bulk read capability**

Simultaneous, unique recognition of several tags which pass by an antenna.

**Bus**

Bus system for data exchange, e.g. between CPU, memory and I/O levels. A bus can consist of several parallel cables for data transmission, addressing, control and power supply.

**Bus cycle time**

Time required for a master to serve all slaves or stations in a bus system, i.e. reading inputs and writing outputs.

**Bus line**

Smallest unit connected to a bus, consisting of a PLC, a coupling element for modules on the bus and a module.

**Bus system**

All units which communicate with one another via a bus.

C

### Capacitive coupling

Electrical capacitive couplings occur between cables with different potentials. Typical sources of interference are, for example, parallel-routed signal cables, contactors and electrostatic discharges.

### Coding elements

Two-piece element for the unambiguous assignment of electronic and base modules.

### Configuration

Systematic arrangement of the I/O-modules of a station.

### CPU

Central Processing Unit. Central unit for electronic data processing, the processing core of the PC.

### CSA

(Chip Sharing Approach): DOT offers different companies in the supply chain the opportunity of storing their data on the chip. It is possible to assign read and write rights on the chip.

D

### DHCP

Client-Server-protocol which reduces the effort of assigning IP addresses or other parameters. Serves for dynamic and automatic configuration of devices.

### Digital

A value (e.g. a voltage) which can adopt only certain statuses within a finite set, mostly defined as 0 and 1.

### DIN

German acronym for German Industrial Standard.

### DOT

(Data on Tag): A lot of product data on the chip so that self-acting local decisions can be made without having to consult the central database.

### DP Master class 1

The automation system (PLC), which mainly implements cyclic data processing. The „DPV1“ functions can be additionally(optionally used.
(DPM1/DPC1 also) .

### DP Master class 2

Acyclic requirement data are transmitted exclusively. The data transfer can be implemented for example by an engineering tool (PC application program.

### DPV1

Function expansion of PROFIBUS-DP. In addition to the cyclic process data, the requirement data can be transmitted via acyclic communication functions. The acyclic services are handled in parallel at the same time and in addition to the cyclic process data transmission with a lower priority.

E

### Earth

All interconnected inactive components that do not take on a dangerous touch potential in the case of a fault.

### Earth strap

Flexible conductor, usually braided, which interconnects inactive parts of the equipment, e.g. the door of a control panel with the control panel chassis.

### EIA

Electronic Industries Association. – association of electrical companies in the United States

### Electrical components

All objects that produce, convert, transmit, distribute or utilize electrical power (e.g. conductors, cable, machines, control devices).

### EMC

Electromagnetic compatibility. The ability of an electrical part to operate in a specific environment without fault and without exerting a negative influence on its environment.

### EPC

(Electronic Product Code)
The actual data for the article/product are stored in one or more databases with manufacturers, forwarders, dealers etc.

### ESD

Electrostatic Discharge.

F

### Fieldbus

Data network on sensor/actuator level. A fieldbus connects the equipment on the field level. Characteristics of a fieldbus are a high transmission security and real-time behaviour.

### Field power supply

Voltage supply for devices in the field as well as the signal voltage.

### Force Mode

Software mode which enables the user to set his plant to a required state by forcing certain variables on the input and output modules.

### Function Code

Are integrated into the data telegram with the Modbus. Contains commands to read and write input and output data, etc.

G

### Galvanic coupling

Galvanic coupling occurs between two cables through which an electrical current is flowing. Typical sources of interference are for example, are stating motors, static discharges, switched devices and a different potential between the component housings and the common power supply.

### GND

GROUND (Potential 0).

### Ground

Expression used in electrical engineering to describe an area whose electrical potential is equal to zero at any given point. In neutral grounding devices, the potential is not necessarily zero, and one speaks of the “ground reference”.

### Ground connection

One or more components that have a good and direct contact to earth.

### Ground reference

Potential of ground in a neutral grounding device. Unlike earth whose potential is always zero, it may have a potential other than zero.

H

### Hexadecimal

System of representing numbers in base 16 with the digits 0…9, and further with the letters A, B, C, D, E and F.

### Hysteresis

A sensor can get caught up at a certain point, and then "waver" at this position. This condition results in the counter content fluctuating around a given value. Should a reference value be within this fluctuating range, then the relevant output would be turned on and off in rhythm with the fluctuating signal.

I

### I/O

Input/Output.

### Impedance

Total effective resistance that a component or circuit has for an alternating current at a specific frequency.

### Inactive metal components

Conductive components that cannot be touched and are electrically isolated from active metal components by insulation, but can adopt voltage in the event of a fault.

### Inductive coupling

Magnetic inductive couplings occur between two cables through which an electrical current is flowing. The magnetic effect caused by the electrical currents induces an interference voltage. Typical sources of interference are for example, transformers, motors, parallel-routed network and HF signal cables.

### Intelligent modules

Intelligent modules are modules with an internal memory, able to transmit certain commands (e.g. substitute values and others).

### IP protocol

Abbreviation for Internet Protocol, protocol for the packet-oriented and connectionless transport of data packets from a transmitter to a receiver crossing different networks.

L

### Low impedance connection

Connection with a low AC impedance.

### LSB

Least Significant Bit. Bit with the lowest significance.

M

### MAC-ID

Manufacturer-specific ID assigned according to a defined key for unique identification of a node in a network.

### Modbus TCP

The Modbus protocol is thus a part of the TCP/IP protocol.

The Modbus communicates with the help of function codes which are integrated into the data telegram. Modbus TCP uses the transport control protocol (TCP) for the transmission of the Modbus application protocol for data transfer in Ethernet-TCP/IP networks.

### Mode

Mode.

### Module bus

The module bus is the internal bus in a BL67 station. The BL67 module uses it to communicate with the gateway. It is independent of the fieldbus.

### MSB

Most Significant Bit. Bit with the highest significance.

O

### Overhead

System administration time which the system requires each time with each transmission cycle.

P

**Parameterisation**

Determination of parameters of the individual bus stations, or their modules via the configuration software of the DP master.

**PIB**

Proxy Ident Function Block. This function block represents the identification system of the control. Thus a common programming interface exists for the actual application.

**Ping**

Implementation of an echo-protocol, used for testing whether a particular host is operating properly and is reachable on the network from the testing host.

**PLC**

Programmable Logic Controller.

**Potential compensation**

The alignment of electrical levels of electrical components and external conductive components by means of an electrical connection.

**Potential free**

Galvanic isolation of the reference potentials in I/O-modules of the control and load circuits.

**Potential linked**

Electrical connection of the reference potentials in I/O-modules of the control and load circuits.

**Protective earth**

Electrical conductor for protection against dangerous shock currents. Generally represented by PE (protective earth).

R

**Reaction time**

The time required in a bus system between a reading operation being sent and the receipt of an answer. It is the time required by an input module to change a signal at its input until the signal is sent to the bus system.

**Reference potential**

Potential from which all voltages of connected circuits are viewed and/or measured.

**Repeater**

Amplifier for signals transmitted via a bus.

**RFID**

Radio frequency identification

**RFID technology**

This technology allows a non-contact transfer of data. The data is transferred on the basis of radio frequency technology. A „Transponder” page 4-6 is used as a data carrier.

**RS 485**

Serial interface in accordance with EIA standards, for fast data transmission via multiple transmitters.

S

**Serial**

Type of information transmission, by which data is transmitted bit by bit via a cable.

**Shield**

Conductive screen of cables, enclosures and cabinets.

### Shielding

Description of all measures and devices used to join installation components to the shield.

### Short-circuit proof

Characteristic of electrical components. A short-circuit proof part withstands thermal and dynamic loads which can occur at its place of installation due to a short circuit.

### Station

A functional unit or I/O components consisting of a number of elements.

T

### Terminating resistor

Resistor on both ends of a bus cable used to prevent interfering signal reflections and which provides bus cable matching. Terminating resistors must always be the last component at the end of a bussegment.

### TCP

Abbreviation for Transmission Control Protocol, connection-oriented transport protocol within the Internet protocol. Certain error detection mechanisms (i.e. acknowledgements, time-out monitoring) can guarantee a safe and error free data transport.

### To ground

Connection of a conductive component with the grounding connection via a grounding installation.

### Topology

Geometrical structure of a network or the circuitry arrangement.

### Transponder

(**Trans**mitter + Res**ponder**)
Response transmitting device. A transponder consists of a microchip (with a unique identification number), a transmitting/receiving antenna and an enclosure. The data is transferred between the reading device and the transponder via electromagnetic waves.

### Transponder technology

(see also „RFID technology” page 4-5)

U

### UDP

User Datagram Protocol. UDP is an transport protocol for the connectionless data between Ethernet hosts.

### UID

Unique Identifier. The UID is a unique transponder serial number. As an address it refers to the corresponding data which can be found on the transponder or the tagged product. This data can, for example, be stored in a database.

TURCK

Industrial
Automation

**www.turck.com**

D101648 0308

**Hans Turck GmbH & Co. KG**
45472 Mülheim an der Ruhr
Germany
Witzlebenstraße 7
Tel. +49 (0) 208 4952-0
Fax +49 (0) 208 4952-264
E-Mail more@turck.com
Internet www.turck.com