

# VPC 3+

# Software Manual

**Revision 6.00**

**The Cl ever Al t er n at i ve**



#### Liability Exclusion

We have tested the contents of this document regarding agreement with the hardware and software described. Nevertheless, there may be deviations and we do not guarantee com- plete agreement. The data in the document is tested periodically, however. Required correc- tions are included in subsequent versions. We gratefully accept suggestions for improve- ment.

#### Copyright

Copyright © profichip GmbH 2001. All Rights Reserved.

Unless permission has been expressly granted, passing on this document or copying it, or using and sharing its content are not allowed. Offenders will be held liable. All rights re- served, in the event a patent is granted or a utility model or design is registered.

This document is subject to technical changes.

[VPC 3+ i](#_TOC_250001)

[Software Manual i](#_TOC_250000)

1. [Table of Contents iii](#_bookmark0)
2. [Introduction 1](#_bookmark1)
   1. [Software package 2](#_bookmark2)
   2. [Software package PA007050 2](#_bookmark3)
   3. [Software package PA007062 2](#_bookmark4)
   4. [Structure of PA007062 / PA007050 software package 3](#_bookmark5)
   5. [PROFIBUS DP 6](#_bookmark6)
   6. [PROFIBUS DPV1 6](#_bookmark7)
   7. [PROFIBUS DPV2 7](#_bookmark8)
   8. [How a PROFIBUS DP Slave Works 7](#_bookmark9)
   9. [Helpful documents 11](#_bookmark10)
   10. [Helpful links 11](#_bookmark11)
3. [Initialization 13](#_bookmark12)
   1. [Configuration of platform.h 13](#_bookmark13)
      1. [Settings of platform.h 13](#_bookmark14)
      2. [Example 8051, KEIL compiler 14](#_bookmark15)
      3. [Example 80165, TASKING compiler 14](#_bookmark16)
      4. [Example AtMega128 15](#_bookmark17)
      5. [Example ARM9, GNU compiler 15](#_bookmark18)
   2. [Configuration of DpCfg.h 16](#_bookmark19)
      1. [Profibus Services 16](#_bookmark20)
      2. [General Slave Parameter 17](#_bookmark21)
      3. [Buffer Initialization 17](#_bookmark22)
      4. [Settings for I&M functionality 18](#_bookmark23)
      5. [Settings for MSAC\_C1 18](#_bookmark24)
      6. [Settings for MSAC\_C1 Alarm 18](#_bookmark25)
      7. [Settings for MSAC\_C2 Service 18](#_bookmark26)
      8. [Settings for Isochron Mode 19](#_bookmark27)
      9. [Settings for DXB Subscriber Mode 19](#_bookmark28)
      10. [Set Hardware Mode 20](#_bookmark29)
   3. [Configuration of dp\_inc.h 27](#_bookmark30)
   4. [Configuration of main.c 28](#_bookmark31)
   5. [Memorytest of VPC3+ 29](#_bookmark32)
   6. [Initializing of VPC3+ 30](#_bookmark33)
   7. [Starting VPC3 31](#_bookmark34)
   8. [Startup Telegram Sequence 32](#_bookmark35)
      1. [Bus monitoring (Startup sequence) 33](#_bookmark36)
4. [General VPC3-DP Functions 37](#_bookmark37)
   1. [Interrupt Indication Function 37](#_bookmark38)
      1. [Reading the Indication 37](#_bookmark39)
      2. [Acknowledging the Indication 39](#_bookmark40)
      3. [Ending the Indication 39](#_bookmark41)
      4. [Polling the Indication 40](#_bookmark42)
   2. [Parameter Data 41](#_bookmark43)
      1. [Checking the Parameter Data 41](#_bookmark44)
      2. [Parameter Data Structure 43](#_bookmark45)
   3. [Configuration Data 48](#_bookmark46)
      1. [Checking Configuration Data 48](#_bookmark47)
      2. [Configuration Data Formats 51](#_bookmark48)
   4. [Transfer of Output Data 53](#_bookmark49)
   5. [Transfer of Input Data 54](#_bookmark50)
   6. [Diagnostic 55](#_bookmark51)
      1. [Transferring Diagnostic Data 55](#_bookmark52)
      2. [Structure of diagnostic block 56](#_bookmark53)
      3. [User specific diagnostic 58](#_bookmark54)
   7. [Changing the Slave Address 61](#_bookmark55)
   8. [Global Control Commands 63](#_bookmark56)
   9. [Watchdog Timeout in DP-Control 65](#_bookmark57)
      1. [Leaving the Data Exchange State 66](#_bookmark58)
   10. [VPC3\_Reset (Go\_Offline) 67](#_bookmark59)
   11. [Leave Master 67](#_bookmark60)
   12. [FatalError (DP+MSAC\_C1+MSAC\_C2) 67](#_bookmark61)
5. [DPV1 Extensions 69](#_bookmark62)
   1. [Functional Description of the DPV1 Services 69](#_bookmark63)
      1. [Initiate (MSAC\_C2) 69](#_bookmark64)
      2. [Abort (MSAC\_C2) 69](#_bookmark65)
      3. [Read (MSAC\_C1 and MSAC\_C2) 69](#_bookmark66)
      4. [Write (MASC\_C1 and MSAC\_C2) 70](#_bookmark67)
      5. [Data Transport (MSAC\_C2) 70](#_bookmark68)
      6. [Diagnosis, Alarms, and Status Messages in the case of](#_bookmark69) [DPV1 70](#_bookmark69)
      7. [Error Handling 70](#_bookmark70)
   2. [Initialization 73](#_bookmark71)
      1. [Settings for DPV1 in the DpCfg.h 73](#_bookmark72)
   3. [DP-V1 Callback Functions 77](#_bookmark73)
      1. [Dpv1\_Msac2InitiateReq (MSAC\_C2) 78](#_bookmark74)
      2. [MSAC\_C2\_INITIATE\_REQ\_TO\_RES (MSAC\_C2) 81](#_bookmark75)
      3. [Dpv1\_Msac2AbortInd 82](#_bookmark76)
      4. [Dpv1\_ReadReq (MSAC\_C1+MSAC\_C2) 84](#_bookmark77)
      5. [DpV1\_WriteReq (MSAC\_C1+MSAC\_C2) 85](#_bookmark78)
      6. [Dpv1\_Msac2DataTransportReq (MSAC\_C2) 86](#_bookmark79)
   4. [DPV1 Alarm-Handling 87](#_bookmark80)
      1. [Coding of the Alarm PDU 87](#_bookmark81)
      2. [Coding of the Status PDU 89](#_bookmark82)
      3. [Example for Ext\_Diag\_Data (Alarm and Status PDU) 91](#_bookmark83)
      4. [Coding of the Alarm\_Ack-PDU 92](#_bookmark84)
      5. [Alarm User Callback Functions 93](#_bookmark85)
6. [DPV2 Services 95](#_bookmark86)
   1. [Isochron Mode (IsoM) 95](#_bookmark87)
      1. [General 95](#_bookmark88)
      2. [Isochron Mode 96](#_bookmark89)
      3. [Poor Sync Mode 98](#_bookmark90)
      4. [Structured Prm-Data for Isochron Mode 100](#_bookmark91)
   2. [Data-eXchange-Broadcast (DXB) 101](#_bookmark92)
      1. [Publisher 101](#_bookmark93)
      2. [Subscriber 102](#_bookmark94)
      3. [Structured PRM-Data: DXB Linktable 102](#_bookmark95)
      4. [Structured PRM-Data: DXB Subscribertable 103](#_bookmark96)
      5. [Structure of VPC3+ DXB-Link Table 103](#_bookmark97)
      6. [Structure of VPC3+ DXB Link Status 104](#_bookmark98)
      7. [Functional Description of the DXB Services 104](#_bookmark99)
7. [Trouble-shooting 109](#_bookmark100)
8. [Appendix 111](#_bookmark101)
   1. [Revision History 111](#_bookmark102)

## Notes:

Profichip’s **VPC3+** is a communication chip with processor interface for intelligent slave applications. VPC3+ handles the complete PROFIBUS- DP/DPV1 slave protocol independently and relieves the application processor of all time critical communication tasks. When VPC3+ carries out a DP communication it automatically sets up all DP-SAPs. All necessary timers and monitoring functions are integrated in the chip. Therefore almost the entire performance of the external controller is available for the application.

The UART converts the asynchronous serial PROFIBUS data stream into internal parallel data or vice-versa. Data is synchronized to system clock and processed by the microsequencer. The VPC3+ is capable of automatically identifying and controlling transmission rates up to 12 Mbit/s. The baudrate-generator derives the transmission clock from the system clock. The IDLE- and SYNI- (synchronization interval) timer observes the correct timing of the DP-telegrams according to the PROFIBUS-DP standards and especially controls the idle time before the next request telegram may occur. In case of timing violations the microsequencer will get a notification. The watchdog-timer observes the entire communication. If the watchdog is not re-triggered within the parameterized time (e.g. if the master application fails), the outputs are switched off automatically.

The 2/4 KByte on-chip communication RAM serves as an interface between the VPC3+ and the software/ application. Various telegram information is made available to the user in separate data buffers. Three input buffers and three outputs are provided for data communication. One buffer is always available for communication. Therefore, no resource problems can occur. For optimal diagnosis support, VPC3+ has two diagnosis buffers, that is, one diagnosis buffer is always assigned to VPC3+.

The microsequencer controls the entire process of PROFIBUS-DP/DPV1 protocol handling. Incoming data handed over by the UART is analyzed according to PROFIBUS-DP. If a service is recognized to be valid, user data is stored in the communication RAM and the interrupt controller generates an indication interrupt. Telegrams having frame errors (e.g. parity- or checksum errors) will be rejected. If the service of the telegram is recognized but its request does not make sense, a corresponding response telegram will be generated automatically. As a result user data will then be rejected to avoid unnecessary resource allocation within the microcontroller. The behavior of the microsequencer can be parameterized via mode- and parameter registers.

The Bus Interface Unit is a configurable synchronous/ asynchronous 8-bit interface for various microcontrollers / processors. The user can directly access the internal RAM or the Parameter Registers via the 11-bit address bus.

## Software package

The VPC3+ program package relieves the user of hardware register manipulations and memory calculations. It also provides a convenient „C“- interface to the DP and handles the completely statemachine for DPV1.

## Software package PA007050

The software package consist of three application demos and is free available ([http://www.profichip.com/products/overviewasics/dp-slave-vpc3-](http://www.profichip.com/products/overviewasics/dp-slave-vpc3-c/dp-v0-firmware/?L=5) [c/dp-v0-firmware/?L=5](http://www.profichip.com/products/overviewasics/dp-slave-vpc3-c/dp-v0-firmware/?L=5) ):

* + - EASY4711:
      * Simple slave with 2 byte of input data and 2 byte of output data
      * Diagnostic: No
    - EASYADAC:
      * Modular slave ( up to 244 modules )
      * Diagnostic: No
    - DPV0AFFE:
      * Modular slave with 6 modules
      * Diagnostic: Yes
        + Modulstatus
        + Identifier related
        + Device related

## Software package PA007062

* + - DPV1AFFE
      * Modular slave with 6 modules
      * Diagnostic: Yes
        + Modulstatus
        + Identifier related
      * DP-V1 functions:
        + Data set read
        + Data set write
        + I&M functions
        + Alarm

Processalarm Diagnosticalarm

## Structure of PA007062 / PA007050 software package

|  |  |  |
| --- | --- | --- |
| **Directory/**  **Sub-Directory** | **File Name** | **Explanation** |
| \DOC\_DIR\ | VPC3+CLF3\_UMxyz.pdf | Manual VPC3+CLF3 |
| VPC3+S\_UMxyz.pdf | Manual VPC3+S |
| VPC3+\_SoftwareDescription\_Vxyz.pdf | Documentation of VPC3+ software |
| PROFIBUS\_Description.pdf | Short PROFIBUS description |
| Diagnosis.pdf | Description of PROFIBUS diagnosis |
| GSD\_Spec\_2122\_V51.pdf | GSD-file description |
| ProfileGuidelines-I&M\_3502.pdf | Description of DP-V1 I&M-functionality |
| \Customer\ |  | Directory of DP-V0 functions: |
| \DPV0\_DRV\ |
| \DPV1\_DRV\ |  | Directory of DP-V1 functions |
| \GSD |  | GSD-file |
|  | Main.c | Main function call |
|  | Platform.h | Microcontroller settings, data types |
|  | DpCfg.h | Configuration file for VPC3+ |
|  | DpAppl.c | Application demo |
|  | DpAppl.h | Structures of application demo |
|  | DpPrm.c | Handling of PROFIBUS Parameter- telegram |
|  | DpCfg.c | Handling of PROFIBUS Configuration- telegram |
|  | DpDiag.c | Handling of PROFIBUS diagnostics |
|  | DpV1.c | Handling of DP-V1 services |
|  | DpIm.c | Handling of I&M functionality |
| \EvalBoard\ |  | PROFICHIP-Evaluation-board |
| \DPV0\_DRV\ | Directory of DP-V0 functions: |
| \DPV1\_DRV\ |  | Directory of DP-V1 functions |
| \GSD |  | GSD-file |
| \Ext\ |  | Directory of ATMEL 8051 microcontroller |
|  | startup.asm | Start routine |
|  | regsnd1.h | Defines T8xC51SND1 components |
|  | extsnd1.h | Extension to regsnd1.h |
|  | DpDebug.h | Header file of debug functions |
|  | DpDebug.c | Debug functions |
|  | Lcd.h | Defines for LCD-display |
|  | Lcd.c | Functions for LCD-display |
|  | Serio.h | Defines for serial functions |
|  | Serio.c | Serial functions |
|  | Twi.h | Defines IIC |
|  | Twi.c | Functions for IIC |
|  | Main.c | Main function call |
|  | Platform.h | Microcontroller settings, data types |
|  | DpCfg.h | Configuration file for VPC3+ |
|  | DpAppl.c | Application demo |
|  | DpAppl.h | Structures of application demo |
|  | DpPrm.c | Handling of PROFIBUS Parameter-  telegram |
|  | DpCfg.c | Handling of PROFIBUS Configuration- telegram |
|  | DpDiag.c | Handling of PROFIBUS diagnostics |
|  | DpV1.c | Handling of DP-V1 services |
|  | DpIm.c | Handling of I&M functionality |
| \Examples\ |  |  |

|  |  |  |
| --- | --- | --- |
| \DPV0AFFE\ |  | DP-V0 example with 6 modules and diagnostics |
| \EASY4711\ |  | DP-V0 example with two byte of input and  two byte of output data, no diagnostic |
| \EASYADAC\ |  | DP-V0 example with 244 byte of input and 244 byte of output data, no diagnostic |
|  | platform\_cust.h | Microcontroller settings, data types for  customer project (parallel mode ) |
|  | platform\_eva.h | Microcontroller settings, data types for profichip evaluation board ( parallel mode ) |
|  | platform\_cust\_ser.h | Microcontroller settings, data types for  customer project (serial mode ) |
|  | platform\_eva\_ser.h | Microcontroller settings, data types for profichip evaluation board ( serial mode ) |
|  | DpCfg\_isr.h | Configuration file for VPC3+ (interrupt  driven) |
|  | DpCfg\_poll.h | Configuration file for VPC3+ (polling mode) |
|  | DpAppl.c | Application demo |
|  | DpAppl.h | Structures of application demo |
|  | DpPrm.c | Handling of PROFIBUS Parameter- telegram |
|  | DpCfg.c | Handling of PROFIBUS Configuration- telegram |
|  | DpDiag.c | Handling of PROFIBUS diagnostics |
| \DPV1\_AFFE\ |  | DP-V1 example with 6 modules, alarms and I&M functionality |
|  | platform\_cust.h | Microcontroller settings, data types for customer project ( parallel mode ) |
|  | platform\_eva.h | Microcontroller settings, data types for profichip evaluation board ( parallel mode ) |
|  | platform\_cust\_ser.h | Microcontroller settings, data types for customer project (serial mode ) |
|  | platform\_eva\_ser.h | Microcontroller settings, data types for profichip evaluation board ( serial mode ) |
|  | DpCfg\_isr.h | Configuration file for VPC3+ (interrupt driven) |
|  | DpCfg\_poll.h | Configuration file for VPC3+ (polling mode) |
|  | DpAppl.c | Application demo |
|  | DpAppl.h | Structures of application demo |
|  | DpPrm.c | Handling of PROFIBUS Parameter- telegram |
|  | DpCfg.c | Handling of PROFIBUS Configuration-  telegram |
|  | DpDiag.c | Handling of PROFIBUS diagnostics |
|  | DpV1.c | Handling of DP-V1 services |
|  | DpIm.c | Handling of I&M functionality |

**Figure 2-1: Content of the directory**

Subdirectory DPV0\_DRV:

|  |  |  |
| --- | --- | --- |
| **Directory/**  **Sub-Directory** | **File Name** | **Explanation** |
| \DPV0\_DRV\ |  | Directory of DP-V0 functions: |
| dp\_if.h | Defines,structures and macros of VPC3+ |
| dp\_if.c | Basic functions for VPC3+ |
| dp\_isr.c | Interrupt, poll routine for VPC3+ |
| dpl\_list.h | Macros for double pointered list |
| dp\_inc.h | Header include hierarchy |

**Figure 2-2: Content of the directory**

Subdirectory DPV1\_DRV:

|  |  |  |
| --- | --- | --- |
| **Directory/**  **Sub-Directory** | **File Name** | **Explanation** |
| \DPV1\_DRV\ |  | Directory of DP-V1 functions |
| dp\_fdl.c | Basic fdl-driver |
| dp\_msac1.c | Driver for acyclic class1 messages |
| dp\_msac2.c | Driver for acyclic class2 messages |

**Figure 2-3: Content of the directory**

**Only in software package PA007062!**

## PROFIBUS DP

PROFIBUS DP was developed for fast, cyclical input and output traffic, with the application emphasis being on the field level. The data traffic in the master-slave method is standardized in the EN 50 170; simple as well as intelligent field devices can be interconnected.

## PROFIBUS DPV1

In many cases, cyclical data exchange according to EN 50 170 is no longer sufficient today for more complex devices. For that reason, it became necessary to define acyclical services as PROFIBUS extensions. These extensions have been defined in the technical guideline of the Profibus Trade Organization (PNO). Field devices can use these services optionally.

Some intelligent field devices need the following:

Gapless reparameterizaton of the application process Free access to any parameters in a field device Transmission of data of variable length



For the sake of simplicity, these services may be transferred to the field devices acyclically, and run parallel to the cyclical data traffic. Standard field devices and devices that need these optional extensions can be operated jointly on the same bus with the functionality that is supported respectively.

The following services are specified as optional services between Class 1 masters and a slave as MSAC\_C1 (Master-Slave acyclic communication Class 1):

Read the data set of a slave (DS\_Read) Write the data set of a slave (DS\_Write) Alarm acknowledgement (Alarm\_Ack)



The following services are specified as optional services between Class 2 masters and a slave as MSAC\_C2 (Master-Slave acyclic communication Class 2):

Initiate



Read the data set of a slave (DS\_Read) Write the data set of a slave (DS\_Write) Transport (Data\_Transport)

Abort

## PROFIBUS DPV2

PROFIBUS DPV2 adds a number of new features to the existing protocol stack to provide for slave-to-slave communications, time synchronization and an isochronal bus cycle. PROFIBUS now has the capability to provide for both acyclic communications via DPV1 and also slave-to-slave communications via DPV2, creating new application areas particularly in motion control (PROFIdrive) and safety (PROFIsafe).

The new functions of DPV2 include the establishment of an isochronous bus cycle (occurring in equal intervals of time) which allows closed-loop control between master and slave devices. With clock deviations of less than 1 microsecond, high-precision positioning can be realized. Slave-to- slave communication decreases the cycle time between master and slave and reduces the response time by 60 – 90 %.

Time synchronization provides a time stamp function so that events can be followed or tracked precisely, easing the registration/detection of timed events and facilitating the diagnosis of malfunctions and the correct chronological planning of actions. With the new upload and download functionality, any size data packet can be loaded into a field device with one command. Program updates or exchange of devices can be carried out without the troublesome and complicated loading processes, which are different for every manufacturer. The transfer into non-volatile storage or the start/stop command for the field device are also supported.

## How a PROFIBUS DP Slave Works

For clarification, the state machine of a DP slave is briefly described below. The state machine regulates the defined, standard-conforming response of a DP slave in the possible situations. A detailed description is provided in the corresponding documents.

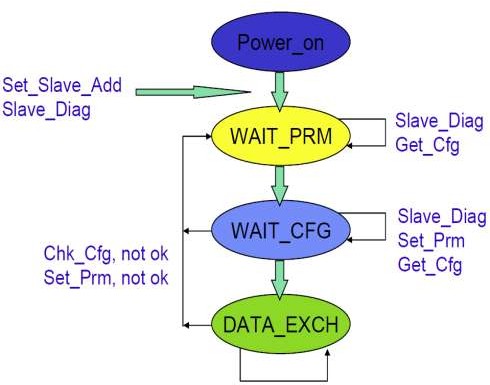
The sequence, in principle, of this state machine is helpful to understanding the firmware sequence. The details are provided in the standard EN 50 170, and the Technical Guidelines. The MSAC\_C2 connection is not interfaced with the cyclical state machine. For that reason, the Class 2 connection is established and cancelled via Initiate and Abort; it is monitored by an idle mechanism.

### Power\_On

A Set\_Slave\_Address message is only accepted in the mode Power\_On.

### Wait\_Prm

After power-up, the slave expects a parameter assignment message. All other types of messages are rejected or are not edited. Data exchange is not yet possible. In the parameter message, at least the information specified by the standard -such as the PNO Ident number, sync/freeze capability, etc.- is stored. In addition, user-specific parameter data is possible. Only the application specifies the meaning of this data. In the configuration of the master interface, certain bits are set, for example, in order to indicate a desired measuring range. The firmware makes this user- specific data available to the application program; the application program evaluates the data; it can accept it or reject it (for example, the desired measuring range can’t be set, and therefore meaningful operation is not possible).



**Figure 2-4 : State Machine**

### Wait\_Cfg

The configuration message specifies the number of input and output bytes. The master informs the slave of how many bytes I/O are being transmitted. The application is informed of the requested configuration for checking. This check results either in a right, a wrong, or an adaptable configuration. If the slave wants to adapt to the desired configuration, a new user data length has to be calculated from the configuration bytes (for example, 4 bytes inputs predefined; only 3 bytes utilized). The application has to decide whether this adaptability is useful. In addition, is possible for each master to poll the configuration of any slave.

### Data\_Exchange

If the firmware as well as the application have accepted the parameter assignment and the configuration as correct, the slave transitions to the mode Data\_Exchange; that is, it exchanges user data with the master.

### Diagnosis

Via the diagnosis, the slave informs the master of its current mode. It consists at least of the information, specified in the standard, in the first six octets, such as the status of the state machine. The user can supplement this information (user diagnosis) with process-specific information (for example, wire break). On the slave’s initiative, the diagnosis can be transmitted as error message and as status message. In addition to three defined bits, the user also influences the application-specific diagnostic data. However, any Master (not only the assigned master) can poll the current diagnostic information.

### Read\_Inputs, Read\_Outputs

Every master can poll the current states of the inputs and outputs of any slave (in the Data\_Exchange mode). The ASIC and the firmware process this function autonomously.

### Watchdog

Along with the parameter message, the slave also receives a watchdog value. If this watchdog is not retriggered through the bus traffic, the state machine transitions to the “safe” state Wait\_Prm.

### MSAC\_C1 (Master Slave Acyclic Communication of Class 1)

The MSAC\_C1 services are used for communicating with a Class 1 master (typically, PLC). These services are available after the master has parameterized and configured the slave; that is, if the slave is in the DataEx mode.

The following services are available:

DS\_READ read data set DS\_WRITE write data set ALARM\_ACK acknowledge alarm



Since these services are permanently coupled to the configuring master C1 and since they run via permanently defined SAPs (50/51), the INITIATE/ABORT/IDLE mechanism is not required. If there is a fault in acyclically data transfer, cyclical communication is influenced also, and vice versa.

### MSAC\_C2 (Master Slave Acyclical Communication of Class 2)

The MSAC\_C2 services are used for communicating with a Class 2 master (typically PC/PG as parameter assignment tool). These services are available immediately after initialization. Since these services are used dynamically, the master has to initiate the establishment of the connection with INITIATE so that the slave can adapt itself to it, and reject the services if necessary (insufficient memory, or no free SAP, …). While the connection is established, both sides monitor the connection with IDLE messages. If the connection is no longer needed, the master or the slave can de- establish the connection by transmitting an ABORT PDU. The IDLE messages are processed within the firmware.

The following services are available:

INITIATE establishment of connection READ read data set



WRITE write data set

DATA\_TRANSPORT general transport service ABORT Cancellation of connection

## Helpful documents

|  |  |  |
| --- | --- | --- |
| **Title** | **Available**  **Language** | **Link** |
| Book:“**The New Rapid Way to PROFIBUS DP**”  Describing PROFIBUS from DP- V0 to DP-V2 | German, English | <http://www.profibus.com/press-> media/pi-books |
| PDF: “**PROFIBUS System**  **Description**” | German, English, Chinese, Japanese, French, Polish,  Russian | <http://www.profibus.com/nc/downlo> ads/downloads/profibus- technology-and-application- system-description/display/ |
| Book: “**PROFIBUS Manual**” A collection of explaining  PROFIBUS networks | German, English | <http://www.profibus.felser.ch/> |

## Helpful links

|  |  |
| --- | --- |
| PROFIBUS international | [http://www.profibus.com](http://www.profibus.com/) |
| PROFIBUS Prof. Max Felser | <http://www.profibus.felser.ch/> |

**Notes:**

## Configuration of platform.h

In order to support the different storage models with some processors, the memory accesses are to be provided with attributes. These attributes depends on the compiler settings.

### Settings of platform.h

|  |  |  |
| --- | --- | --- |
| #define VPC3\_SERIAL\_MODE | 0: VPC3+ works with parallel data bus  1: VPC3+S works with SPI, IIC or PORT\_PIN-mode | |
| #define MOTOROLA\_MODE | 0: the VPC3+ configuration pin MOT/XINT is set to parallel interface INTEL format  1:the VPC3+ configuration pin MOT/XINT is set to parallel interface MOTOROLA format | |
| #define TRUE | 1 |  |
| #define FALSE | !(TRUE) |  |
| #define BOOL | unsigned char | 1 bit basic type |
| #define uint8\_t | unsigned char | 8 bit basic type |
| #define uint16\_t | unsigned int | 16 bit basic type |
| #define uint32\_t | unsigned long | 32 bit basic type |
| #define PTR\_ATTR | xdata | Memory model attribute of VPC3+. ( xdata, near, far, huge ... ) |
| #define VPC3\_PTR | PTR\_ATTR \* | VPC3 Pointer attribut |
| #define VPC3\_ADR | uint16\_t | Attribute of the asic address. ( uint16\_t,  uint32\_t ) |
| #define VPC3\_UNSIGNED8\_PTR | uint8\_t PTR\_ATTR \* | Pointer of byte to VPC3+ |
| #define NULL\_PTR | (void VPC3\_PTR)0 | Zero-pointer |
| #define MEM\_ATTR | xdata | Memory model attribute of local memory.( xdata, near, far, huge ... ) |
| #define MEM\_PTR | MEM\_PTR\_ATTR \* | Pointer attribut of local memory |
| #define MEM\_UNSIGNED8\_PTR | uint8\_t MEM\_PTR\_ATTR \* | Pointer of byte to local memory |
| #define ROM\_CONST | code | Attribute of constant variables. |
| #define \_PACKED\_ |  | Feed a keyword for packing structures. |
| #define LITTLE\_ENDIAN | 0 | 0: deactivated, 1:activated |
| #define BIG\_ENDIAN | 1 | 0: deactivated, 1:activated |
| #define VPC3\_ASIC\_ADDRESS | ((unsigned char \*)0x28000) | VPC3+ address |

**Figure 3-1 : platform.h settings**

### Example 8051, KEIL compiler

|  |  |
| --- | --- |
| #define TRUE | 1 |
| #define FALSE | !(TRUE) |
| #define BOOL | unsigned char |
| #define uint8\_t | unsigned char |
| #define uint16\_t | unsigned int |
| #define uint32\_t | unsigned long |
| #define PTR\_ATTR | xdata |
| #define VPC3\_PTR | PTR\_ATTR \* |
| #define VPC3\_ADR | uint16\_t |
| #define VPC3\_UNSIGNED8\_PTR | uint8\_t PTR\_ATTR \* |
| #define NULL\_PTR | (void VPC3\_PTR)0 |
| #define MEM\_ATTR | xdata |
| #define MEM\_PTR | MEM\_PTR\_ATTR \* |
| #define MEM\_UNSIGNED8\_PTR | uint8\_t MEM\_PTR\_ATTR \* |
| #define ROM\_CONST | code |
| #define \_PACKED\_ |  |
| #define LITTLE\_ENDIAN | 0 |
| #define BIG\_ENDIAN | 1 |
| #define VPC3\_ASIC\_ADDRESS | ((unsigned char \*)0x28000) |

**Figure 3-2 : Example 8051**

### Example 80165, TASKING compiler

|  |  |
| --- | --- |
| #define TRUE | 1 |
| #define FALSE | !(TRUE) |
| #define BOOL | unsigned char |
| #define uint8\_t | unsigned char |
| #define uint16\_t | unsigned int |
| #define uint32\_t | unsigned long |
| #define PTR\_ATTR | far |
| #define VPC3\_PTR | PTR\_ATTR \* |
| #define VPC3\_ADR | uint32\_t |
| #define VPC3\_UNSIGNED8\_PTR | uint8\_t PTR\_ATTR \* |
| #define NULL\_PTR | (void VPC3\_PTR)0 |
| #define MEM\_ATTR | near |
| #define MEM\_PTR | MEM\_PTR\_ATTR \* |
| #define MEM\_UNSIGNED8\_PTR | uint8\_t MEM\_PTR\_ATTR \* |
| #define ROM\_CONST | const |
| #define \_PACKED\_ |  |
| #define LITTLE\_ENDIAN | 1 |
| #define BIG\_ENDIAN | 0 |
| #define VPC3\_ASIC\_ADDRESS | 0x18000 |

**Figure 3-3 : Example 80165**

### Example AtMega128

|  |  |
| --- | --- |
| #define TRUE | 1 |
| #define FALSE | !(TRUE) |
| #define BOOL | unsigned char |
| #define uint8\_t | unsigned char |
| #define uint16\_t | unsigned int |
| #define uint32\_t | unsigned long |
| #define PTR\_ATTR |  |
| #define VPC3\_PTR | PTR\_ATTR \* |
| #define VPC3\_ADR | uint16\_t |
| #define VPC3\_UNSIGNED8\_PTR | uint8\_t PTR\_ATTR \* |
| #define NULL\_PTR | (void VPC3\_PTR)0 |
| #define MEM\_ATTR |  |
| #define MEM\_PTR | MEM\_PTR\_ATTR \* |
| #define MEM\_UNSIGNED8\_PTR | uint8\_t MEM\_PTR\_ATTR \* |
| #define ROM\_CONST | const |
| #define \_PACKED\_ |  |
| #define LITTLE\_ENDIAN | 0 |
| #define BIG\_ENDIAN | 1 |
| #define VPC3\_ASIC\_ADDRESS | ((unsigned char \*)0x8000) |

**Figure 3-4 : Example AtMega128**

### Example ARM9, GNU compiler

|  |  |
| --- | --- |
| #define TRUE | 1 |
| #define FALSE | !(TRUE) |
| #define BOOL | unsigned char |
| #define uint8\_t | unsigned char |
| #define uint16\_t | unsigned short |
| #define uint32\_t | unsigned long |
| #define PTR\_ATTR |  |
| #define VPC3\_PTR | PTR\_ATTR \* |
| #define VPC3\_ADR | uint32\_t |
| #define VPC3\_UNSIGNED8\_PTR | uint8\_t PTR\_ATTR \* |
| #define NULL\_PTR | (void VPC3\_PTR)0 |
| #define MEM\_ATTR |  |
| #define MEM\_PTR | MEM\_PTR\_ATTR \* |
| #define MEM\_UNSIGNED8\_PTR | uint8\_t MEM\_PTR\_ATTR \* |
| #define ROM\_CONST | const |
| #define \_PACKED\_ | attribute ( ( packed ) ) |
| #define LITTLE\_ENDIAN | 1 |
| #define BIG\_ENDIAN | 0 |
| #define VPC3\_ASIC\_ADDRESS | 0x40000000 |

**Figure 3-5 : Example ARM9**

## Configuration of DpCfg.h

The different PROFIBUS services and their parameter defines the user in the file “DpCfg.h”.

### Profibus Services

The user connects the different services via #define in “DpCfg.h”, so that the program code is adapted to the required services respectively.

|  |  |
| --- | --- |
| **Service** |  |
| #define DP\_MSAC\_C1 | 1: Activation of the functionality for the expansion services of the Class 1 master.  0: not activated |
| #define DP\_MSAC\_C2 | 1: Activation of the functionality for the expansion services of the Class 2 master.  0: not activated |
| #define DP\_ALARM | 1: Activation of the functionality for the expansion services of the alarm mode.  0: not activated |
| #define DPV1\_IM\_SUPP | 1: Activation of the functionality for the expansion services of the I&M functionality.  0: not activated |
| #define DP\_SUBSCRIBER | 1: Activation of the functionality for the expansion services of the DXB subscriber mode.  0: not activated |
| #define DP\_TIMESTAMP | 1: Activation of the functionality for the expansion services of the timestamp mode.  0: not activated |
| #define DP\_ISOCHRONOUS\_MODE | 1: Activation of the functionality for the expansion services of the isochronous mode.  0: not activated |

**Figure 3-6 : PROFIBUS Services**

### General Slave Parameter

|  |  |  |
| --- | --- | --- |
| **General Slave Parameter** |  |  |
| #define DP\_ADDR | uint8\_t | PROFIBUS DP-Slave Address (1..125) |
| #define IDENT\_NR | uint16\_t | PROFIBUS Ident Number |
| #define USER\_WD | uint16\_t | User Watchdog |

**Figure 3-7 : General Slave Parameter**

The **ident number** is used for clearly identifying the slave and is included with each diagnostic message from the slave to the master. Request your own number (www.profibus.com).

The **user watchdog** provides that, if the connected microcontroller fails, the VPC3+ leaves the Data Exchange mode after a defined number of data-exchange messages. As long as the microcontroller doesn’t crash, it has to retrigger this watchdog.

### Buffer Initialization

The user must enter the length of the exchange buffers for the different messages in the VPC3+ structure. These lengths determine the data buffers setup in the ASIC, and therefore are dependent in total sum on the ASIC memory.

|  |  |  |
| --- | --- | --- |
| **Buffer** |  |  |
| #define DIN\_BUFSIZE | uint8\_t | Length of the DIn Buffer (0..244 Bytes) |
| #define DOUT\_BUFSIZE | uint8\_t | Length of the DOut Buffer (0..244 Bytes) |
| #define PRM\_BUFSIZE | uint8\_t | Length of the Parameter Buffer (7..244 Bytes) |
| #define DIAG\_BUFSIZE | uint8\_t | Length of the Diagnosis Buffer (6..244 Bytes) |
| #define CFG\_BUFSIZE | uint8\_t | Length of the Configuration Buffer (1..244 Bytes) |
| #define SSA\_BUFSIZE | uint8\_t | Length of the Input Data in the Set\_Slave\_Address- Buffer 0 and 4..244 Bytes |

**Figure 3-8 : Buffer Initialization**

Specifying length 0 for the Set-Slave-Address buffer disables this utility.

### Settings for I&M functionality

|  |  |  |
| --- | --- | --- |
| **I&M** |  |  |
| #define MANUFACTURER\_ID | uint16\_t | Manufacturer ID, order from [www.profibus.com](http://www.profibus.com/) |
| #define IM1\_SUPP |  | 0: service is deactivated  1: service is activated |
| #define IM2\_SUPP |  | 0: service is deactivated  1: service is activated |
| #define IM3\_SUPP |  | 0: service is deactivated  1: service is activated |
| #define IM4\_SUPP |  | 0: service is deactivated  1: service is activated |

**Figure 3-9 : Settings for I&M**

### Settings for MSAC\_C1

|  |  |  |
| --- | --- | --- |
| **Settings for MSAC\_C1 Service** |  |  |
| #define C1\_LEN | uint8\_t | Length of MSAC\_C1 Data (4..244 Bytes) |

**Figure 3-10 : Settings for MSAC\_C1**

### Settings for MSAC\_C1 Alarm

|  |  |
| --- | --- |
| **Settings for MSAC\_C1 Alarm** |  |
| #define DP\_ALARM\_OVER\_SAP50 | 1: The master handles the Alarm Acknowledge over SAP 50  0: The master handles the Alarm Acknowledge over SAP 51 |

**Figure 3-11 : Settings for MSAC\_C1\_Alarm**

### Settings for MSAC\_C2 Service

|  |  |  |
| --- | --- | --- |
| **Settings for MSAC\_C2 Service** |  |  |
| #define DP\_MSAC\_C2\_Time |  | Enables time control for C2 services |
| #define C2\_NUM\_SAPS | uint8\_t | Number of SAPs that the firmware makes available for MSAC\_C2 Connections |
| #define C2\_LEN | uint8\_t | MSAC\_C2 PDU length of the C2-SAP (20...244) |
| #define | uint8\_t | = 0x01 (MSAC\_C2\_READ and |

|  |  |  |
| --- | --- | --- |
| C2\_FEATURES\_SUPPORTED\_ 1 |  | MSAC\_C2\_WRITE supported) |
| #define C2\_FEATURES\_SUPPORTED\_ 2 | uint8\_t | = 0x00 |
| #define C2\_PROFILE\_FEATURES\_1 | uint8\_t | Profile or vendor specific |
| #define C2\_PROFILE\_FEATURES\_2 | uint8\_t | Profile or vendor specific |
| #define C2\_PROFILE\_NUMBER | uint16\_t | Profile or vendor specific |

**Figure 3-12 : Settings for MSAC\_C2 Service**

### Settings for Isochron Mode

|  |  |  |
| --- | --- | --- |
| **Settings for Isochron Mode** |  |  |
| #define SYNCH\_PULSEWIDTH | uint8\_t | Width of synch pulse in 1/12µs |

**Figure 3-13 : Settings for Isochron Mode**

### Settings for DXB Subscriber Mode

|  |  |  |
| --- | --- | --- |
| **Settings for DXB Subscriber Mode** |  |  |
| #define MAX\_LINK\_SUPPORTED | uint8\_t | Number of Links |
| #define MAX\_DATA\_PER\_LINK | uint8\_t | maximal Number of Data per Link |

**Figure 3-14 : Settings for DXB Subscriber Mode**

**PROFICHIPS asics support only one DxB connection!**

### Set Hardware Mode

Next, the user has to configure the hardware function and telegram processing in the Mode Register 0 and 2 of the VPC3+:

#### Changes in Mode Register 0 and 2 are only allowed during start-up, when the VPC3+ is ‘offline’.

|  |  |  |
| --- | --- | --- |
| **Settings for Hardware Mode** |  |  |
| #define INIT\_VPC3\_MODE\_REG\_L | uint8\_t | Mode Register 0 (LowByte) |
| #define INIT\_VPC3\_MODE\_REG\_H | uint8\_t | Mode Register 0 (HighByte) |
| #define INIT\_VPC3\_MODE\_REG\_2 | uint8\_t | Mode Register 2 |
| #define INIT\_VPC3\_MODE\_REG\_3 | uint8\_t | Mode Register 3 |
| #define INIT\_VPC3\_MODE\_IND\_L | uint8\_t | Interrupt Indication (LowByte) |
| #define INIT\_VPC3\_MODE\_IND\_H | uint8\_t | Interrupt Indication (HighByte) |

**Figure 3-15 : Settings for Hardware Mode**

### ModeRegister0

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Address** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 06H  (Intel) | Freeze\_ Supported | Sync\_ Supported | Early\_Rdy | Int\_Pol | MinTSDR | WD\_Base | Dis\_Stop\_ Control | Dis\_Start\_ Control | Mode Reg 0  7 .. 0  See below for coding |

|  |  |
| --- | --- |
|  | **Mode Register 0, Low-Byte, Address 06H (Intel):** |
| Bit 7 | **Freeze\_Supported**: Freeze\_Mode support  0 = Freeze\_Mode is not supported.  1 = Freeze\_Mode is supported |
| Bit 6 | **Sync\_Supported:** Sync\_Mode support  0 = Sync\_Mode is not supported.  1 = Sync\_Mode is supported. |
| Bit 5 | **Early\_Rdy**: Early Ready  0 = Normal Ready: Ready is generated when data is valid (read) or when data has been accepted (write).  1 = Ready is generated one clock pulse earlier. |
| Bit 4 | **INT\_Pol:** Interrupt Polarity  0 = The interrupt output is low-active.  1 = The interrupt output is high-active. |
| Bit 3 | **MinTSDR:** Default setting for the MinTSDR after reset for DP operation or combi operation.  0 = Pure DP operation (default configuration!) |
| Bit 2 | **WD\_Base:** Watchdog Time Base  0 = Watchdog time base is 10 ms (default state)  1 = Watchdog time base is 1 ms |
| Bit 1 | **Dis\_Stop\_Control:** Disable Stopbit Control  0 = Stop bit monitoring is enabled.  1 = Stop bit monitoring is switched off  A Set-Param telegram overwrites this memory cell in the DP mode. (Refer to the user specific data.) |
| Bit 0 | **Dis\_Start\_Control:** Disable Startbit Control  0 = Monitoring the following start bit is enabled.  1 = Monitoring the following start bit is switched off  A Set-Param telegram overwrites this memory cell in the DP mode. (Refer to the user specific data.) |

**Figure 3-16 : Coding of Mode Register 0, Low-Byte**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Address** | **Bit Position** | | | | | | | | **Designation** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| 07H  (Intel) | Reserved | PrmCmd\_ Supported | Spec\_Clear\_ Mode | Spec\_Prm\_ Buf\_Mode | Set\_Ext\_Prm  \_Supported | User\_Time\_ Base | EOI\_Time\_ Base | DP\_Mode | Mode Reg 0  15 .. 8  See below for coding |

|  |  |
| --- | --- |
|  | **Mode Register 0, High-Byte, Address 07H (Intel):** |
| Bit 15 | **Reserved** |
| Bit 14 | **PrmCmd\_Supported**: PrmCmd support for redundancy  0 = PrmCmd is not supported.  1 = PrmCmd is supported |
| Bit 13 | **Spec\_Clear\_Mode:** Special Clear Mode (Fail Safe Mode)  0 = No special clear mode.  1 = Special clear mode. VPC3+ will accept data telegrams with data unit = 0 |
| Bit 12 | **Spec\_Prm\_Buf\_Mode:** Special Parameter Buffer Mode  0 = No special parameter buffer.  1 = Special parameter buffer mode. Parameterization data will be stored directly in the special parameter buffer. |
| Bit 11 | **Set\_Ext\_Prm\_Supported:** Set\_Ext\_Prm telegram support  0 = SAP 53 is deactivated  1 = SAP 53 is activated |
| Bit 10 | **\*)User\_Time\_Base:** Timebase of the cyclical User\_Time\_Clock-Interrupt  0 = The User\_Time\_Clock-Interrupt occurs every 1 ms.  1 = The User\_Time\_Clock-Interrupt occurs every 10 ms. (mandatory DPV1) |
| Bit 9 | **EOI\_Time\_Base:** End-of-Interrupt Timebase  0 = The interrupt inactive time is at least 1 µsec long.  1 = The interrupt inactive time is at least 1 ms long |
| Bit 8 | **DP\_Mode:** DP\_Mode enable  0 = DP\_Mode is disabled.  1 = DP\_Mode is enabled. VPC3+ sets up all DP\_SAPs (default configuration!) |

**Figure 3-17 : Coding of Mode Register 0, High-Byte**

\*) The User\_Time\_Clock is a timer that is used for the timeouts of the MSAC\_C2 connection. It generates a timer tick of 1ms or 10 ms that causes an interrupt if enabled. **The timer has to be set to 10ms if DP\_MSAC\_C2 is defined!** However, the user can attach himself to the timer interrupt routine for his own purposes. If the macro DP\_MSAC\_C2 is not defined, the timer is freely available.

### ModeRegister2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Address** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Reset Value |
| 0CH | 4kB\_Mode | No\_Check\_ Prm\_Reserved | SYNC\_Pol | SYNC\_Ena | DX\_Int\_Port | DX\_Int\_Mode | No\_Check\_ GC\_Reserved | New\_GC\_ Int\_Mode | Mode Reg 2  7 .. 0 |

|  |  |
| --- | --- |
|  | **Mode Register 2, Address 0CH:** |
| Bit 7 | **4kB\_Mode:** Size of internal RAM  0 = 2kB RAM (default).  1 = 4kB RAM |
| bit 6 | **No\_Check\_Prm\_Reserved:** Disables checking of the reserved Prm bits  0 = Reserved bits of Prm-telegram are checked (default).  1 = Reserved bits of Prm-telegram are not checked. |
| bit 5 | **SYNC\_Pol:** Polarity of SYNC pulse (for Isochron Mode only)  0 = negative polarity of SYNC pulse (default)  1 = positive polarity of SYNC pulse |
| bit 4 | **SYNC\_Ena:** Enable generation of SYNC pulse (for Isochron Mode only)  0 = SYNC pulse generation is disabled (default).  1 = SYNC pulse generation is enabled. |
| bit 3 | **DX\_Int\_Port:** Port mode for Dataexchange Interrupt  0 = DX Interrupt not assigned to port DATA\_EXCH (default).  1 = DX Interrupt (synchronized to GC-SYNC) assigned to port DATA\_EXCH. |
| bit 2 | **DX\_Int\_Mode:** Mode of Dataexchange Interrupt  0 = DX Interrupt only generated, if DOUT length not 0 (default).  1 = DX Interrupt generated after every DX-telegram |
| bit 1 | **No\_Check\_GC\_Reserved:** Disables checking of the reserved GC bits  0 = Reserved bits of GC-telegram are checked (default).  1 = Reserved bits of GC-telegram are not checked. |
| bit 0 | **GC\_Int\_Mode:** Controls generation of GC Interrupt  0 = GC Interrupt is only generated, if changed GC telegram is received  1 = GC Interrupt is generated after every GC telegram (default) |

**Figure 3-18 : Coding of Mode Register 2**

### ModeRegister3

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Address** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Reset Value |
| 12H | Reserved | Reserved | Reserved | Reserved | PLL\_Supported | En\_Chk\_SSAP | DX\_Int\_Mode\_2 | GC\_Int\_Mode\_Ext | Mode Reg 3  7 .. 0 |

|  |  |
| --- | --- |
|  | **Mode Register 3, Address 12H:** |
| Bit 7 | **Reserved** |
| bit 6 | **Reserved** |
| bit 5 | **Reserved** |
| bit 4 | **Reserved** |
| bit 3 | **PLL\_Supported:** Enables IsoM-PLL  0 = PLL is disabled.  1 = PLL is enable; For use of PLL, SYNC\_Ena must be set. |
| bit 2 | **En\_Chk\_SSAP:** Evaluation of Source Address Extension  0 = VPC3+ accept any value of S\_SAP  1 = VPC3+ only process the received telegram if the S\_SAP match to the default values represented by the IEC 61158 |
| bit 1 | **DX\_Int\_Mode\_2:** Mode of DX\_Out interrupt  0 = DX\_Out interrupt is generated after each Data\_Exch telegram  1 = DX\_Out interrupt is only generated, if received data is not equal to current data in Dx\_Out buffer of user |
| bit 0 | **GC\_Int\_Mode\_Ext:** extend GC\_Int\_Mode, works only if GC\_Int\_Mode=0  0 = GC Interrupt is only generated, if changed GC telegram is received  1 = GC Interrupt is only generated, if GC telegram with changed Control\_Command is received. |

**Figure 3-19 : Coding of Mode Register 3**

### Activating the Indication Function

The user activates or deactivates interrupts by setting or clearing the corresponding bit in the Interrupt Mask Register. If a bit is set, the corresponding interrupt is disabled (interrupt masked).

#### Masking of an already active interrupt is not possible, that is, an active interrupt remains active after masking, but further activation of this interrupt is rejected.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Address** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 04H  (Intel) | DXB\_Out | New\_Ext\_Prm | DXB\_Link\_Error | User\_Timer\_ Clock | WD\_DP\_ Mode\_Timeout | Baud\_Rate\_ Detect | Go/Leave Data\_EX | MAC\_Reset/ Clock\_Sync | Int-Mask-Reg 7 .. 0  See below for coding |

|  |  |
| --- | --- |
|  | **Interrupt-Mask-Register, Low-Byte, Address 04H (Intel):** |
| Bit 7 | **DXB\_Out:**  VPC 3+ has received a ‘DXB telegram’ and made the new output data available in the ‘N’ buffer. |
| Bit 6 | **New\_Ext\_Prm\_Data:**  The VPC 3+ has received a ‘Set\_Ext\_Param telegram’ and made the data available in the Prm buffer. |
| Bit 5 | **DXB\_Link\_Error:**  The Watchdog cycle is elapsed and at least one Publisher-Subscriber connection breaks down. |
| Bit 4 | **User\_Timer\_Clock:**  The time base for the User\_Timer\_Clocks has run out ( 1 /10ms). |
| Bit 3 | **WD\_DP\_Control\_Timeout:**  The watchdog timer has run out in the ‘DP\_Control’ WD state |
| Bit 2 | **Baudrate\_Detect:**  The VPC3+ has left the ‘Baud\_Search state’ and found a baud rate. |
| Bit 1 | **Go/Leave\_DATA\_EX:**  The DP\_SM has entered or exited the ‘DATA\_EX’ state |
| Bit 0 | **MAC\_Reset (used if CS\_Supported=0):**  After it processes the current request, the VPC3+ has arrived at the offline state (by setting the ‘Go\_Offline bit’)  **Clock\_Sync (used if CS\_Supported=1):**  The VPC3+ has received a Clock\_Value telegram or an error occurs. Further differentiation is made in the Clock\_Sync\_buffer. |

**Figure 3-20 : Interrupt Mask Register, Low-Byte**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Address** | **Bit Position** | | | | | | | | **Designation** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| 05H  (Intel) | FDL\_ind | Poll\_End\_ind | DX\_Out | Diag\_Buffer\_  Changed | New\_Prm\_  Data | New\_Cfg\_  Data | New\_SSA\_  Data | New\_GC  Command | Int-Mask-Reg 15 .. 8  See below for coding |

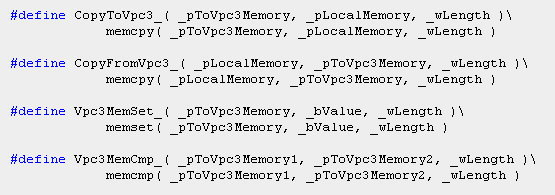
|  |  |
| --- | --- |
|  | **Interrupt Mask Register 0, High-Byte, Address 05H (Intel):** |
| Bit 15 | **FDL\_Ind:**  The VPC 3+ has received an acyclic service request and made the data available in an indication buffer. |
| Bit 14 | **Poll\_End\_Ind:**  The VPC 3+ has sent the response to an acyclic service. |
| Bit 13 | **DX\_Out:**  0 = No special clear mode.  1 = Special clear mode. VPC3+ will accept data telegrams with data unit = 0 |
| Bit 12 | **Diag\_Buffer\_Changed:**  Due to the request made by ‘New\_Diag\_Cmd,’ VPC3+ exchanged the diagnostics buffer and again made the old buffer available to the user. |
| Bit 11 | **New\_Prm\_Data:**  The VPC3+ has received a ‘Set\_Param telegram’ and made the data available in the Prm buffer. |
| Bit 10 | **New\_Cfg\_Data:**  The VPC3+ has received a ‘Check\_Cfg telegram’ and made the data available in the Cfg buffer. |
| Bit 9 | **New\_SSA\_Date:**  The VPC3+ has received a ‘Set\_Slave\_Address telegram’ and made the data available in the SSA buffer. |
| Bit 8 | **New\_GC\_Command:**  The VPC3+ has received a ‘Global\_Control telegram’ and this byte is stored in the ‘R\_GC\_Command’ RAM cell. |

**Figure 3-21 : Interrupt Mask Register, High-Byte**

For test purpose, the user can trigger any interrupt by writing to the Inter- rupt Request Register.

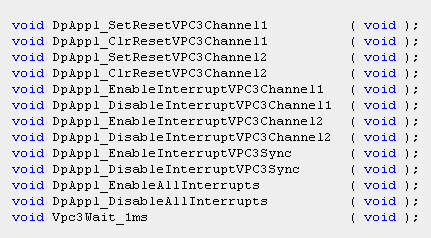
## Configuration of dp\_inc.h

In the dp\_inc.h header file all external functions for the handling of VPC3+ are defined. The user can adapt here own functions for copying data to the VPC3+ or for copying data from the VPC3+.

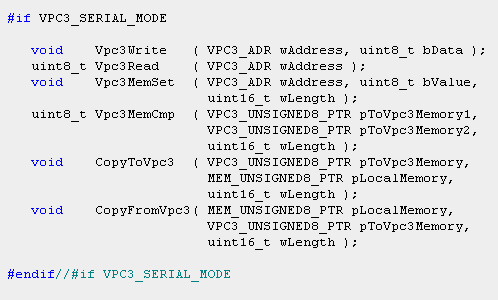


## Configuration of main.c

The user must add own code to following functions:



If the VPC3+S is setup to serial mode, following functions must be added:



## Memorytest of VPC3+

Before initialization of VPC3+ the memory of VPC3+ should be checked.

|  |  |  |
| --- | --- | --- |
| **DP\_ERROR\_CODE VPC3\_MemoryTest( void )** | | |
| Function | This function checks the memory of VPC3+. The starting address is 16hex and the end address depends on DP\_VPC3\_4KB\_MODE (DpCfg.h). | |
| Parameter |  |  |
| Return Value | DP\_OK DP\_VPC3\_ERROR | Memory test OK Memory test failed |

**Figure 3-22 : Function VPC3\_MemoryTest()**

#### Memory test failed:

In this case read the status register of VPC3+. This register has a default value.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Address** | **Bit Position** | | | | | | | | **Designation** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| 05H  (Intel) | VPC 3+ Release | | | | Baudrate | | | | Status-Reg |
| 3 | 2 | 1 | 0 | 3 | 2 | 1 | 0 |

|  |  |
| --- | --- |
| **Status Register, High-Byte, Address 05H (Intel):** | |
| bit 15-12 | **VPC 3+-Release 3..0 :** Release number for VPC 3+  0000 = VPC3+  1011 = VPC3+B  1100 = VPC3+C  1101 = MPI12x  1110 = VPC3+S  Rest = Not possible |
| bit 11-8 | **Baudrate 3..0 :** The baudrate found by VPC 3+  0000 = 12,00 Mbit/s  0001 = 6,00 Mbit/s  0010 = 3,00 Mbit/s  0011 = 1,50 Mbit/s  0100 = 500,00 Kbit/s  0101 = 187,50 Kbit/s  0110 = 93,75 Kbit/s  0111 = 45,45 Kbit/s  1000 = 19,20 Kbit/s  1001 = 9,60 Kbit/s  1111 = after reset and during baudrate search Rest = not possible |

**Figure 3-23: Status Register, High-Byte**

## Initializing of VPC3+

The function **VPC3\_Initialization()** handles the completely initializing of the VPC3+.

Initializing RAM to zero Calculating buffer structures



Initializing the ASIC with DP and FDL if necessary

If necessary: setting up the MSAC\_C2 SAPs according to transfer parameters. The MSAC\_C1 SAPs mentioned above are set up, but are not yet opened.

Initializing the resource manager (RM) and setting up the RM SAP. The RM SAP will only be opened after the ASIC is started with DPSE\_START. The MSAC\_C2 services are available immediately after DPSE\_START.

Enter the first free SAP as response data for RM SAP.

|  |  |  |
| --- | --- | --- |
| **DP\_ERROR\_CODE VPC3\_Initialization( uint8\_t bSlaveAddress, uint16\_t wIdentNumber, CFG\_STRUCT sCfgData )** | | |
| Function | Initialization of VPC3+ | |
| Parameter | bSlaveAddress | Address of the slave |
| wIdentNumber | PROFIBUS Ident Number |
| sCfgData | Default configuration of the slave |
| Return Value | DP\_OK  **\*DP\_NOT\_OFFLINE\_ERROR** | Initialization OK  **\*Error VPC3 is not in OFFLINE state** |
|  | DP\_ADDRESS\_ERROR | Error, DP Slave address |
|  | DP\_CALCULATE\_IO\_ERROR | Error with configuration bytes |
|  | DP\_DOUT\_LEN\_ERROR | Error with Dout length |
|  | DP\_DIN\_LEN\_ERROR | Error with Din length |
|  | DP\_DIAG\_LEN\_ERROR | Error with diagnostics length |
|  | DP\_PRM\_LEN\_ERROR | Error with parameter assignment data length |
|  | DP\_SSA\_LEN\_ERROR | Error with address data length |
|  | DP\_CFG\_LEN\_ERROR | Error with configuration data length |
|  | DP\_LESS\_MEM\_ERROR | Error Overall, too much memory used |
|  | DP\_LESS\_MEM\_FDL\_ERROR | Error Overall, too much memory used |

**Figure 3-24 : Function VPC3\_Initialization()**

**\*If the VPC3+ not in the “OFFLINE” state, reset the VPC3+ once more!**

Before call up the **VPC3\_Initialization()** function the user has to define the default configuration over the structure CFG\_STRUCT. The default configuration will be placed into Read-Configuration buffer.

For example:

typedef struct

{

uint8\_t bLength;

uint8\_t abData[CFG\_BUFSIZE];

} CFG\_STRUCT;

// defined in dp\_if.h

CFG\_STRUCT sRealCfg;

sRealCfg.bLength sRealCfg.abData[0] sRealCfg.abData[1]

= 0x02; // length of configuration data

= 0x25; // master to slave (6Byte)

= 0x17; // slave to master (8Byte)

bError = VPC3\_Initialization( 0x05, 0xADAC, sRealCfg );

## Starting VPC3

If the ASIC could be correctly initialized with **VPC3\_Initialization()**, it still has to be started. Between initialization and start, the user can still initialize buffers in the ASIC.

The VPC3+ goes online with the command:

|  |  |  |
| --- | --- | --- |
| **VPC3\_Start()** | | |
| Function | Starts the VPC3+ | |
| Parameter | None | |
| Return Value | None |  |

**Figure 3-25 : Function VPC3\_Start()**

After the command VPC3\_Start() the VPC3+ generates one DxOut-event to clear the output data..

## Startup Telegram Sequence

### Parameterization- Master

**DP-Slave**

**(Firmware)**

**...**

**(PROFIBUS)**

**...**

**Slave\_Diag (Request) Slave\_Diag (Request)**

**:**

**Slave\_Diag (Request) Slave\_Diag (Request) Slave\_Diag (Response) Set\_Param**

**SC (short acknowledge) Check\_Cfg**

**SC (short acknowledge) Slave\_Diag (Request) Slave\_Diag (Response)**

**:**

**Slave\_Diag (Request) Slave\_Diag (Response) DataExchange (Outputs) DataExchange (Inputs) DataExchange (Outputs) DataExchange (Inputs)**

**:**

**time**

**VPC3 initialization START\_VPC3()**

**:**

**Interrupt(PrmData)**

**Interrupt(CfgData)**

after SetPrmDataOK and SetCfgDataOK the VPC3 goes into the state DataExchange

**Interrupt(DxOut) or**

**Polling(DxOut)**

**:**

**Figure 3-26 : Startup Telegram Sequence**

### Bus monitoring (Startup sequence)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Frame** | **Addr** | **Service** | **Msg type** | **Req/Res** | **SAPS** | **Datalen** | **Data** |
| SD2 | 2->7 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |
| SD2 | 2<-7 | DL | Get Diagnostics | Res | 62<-60 | 6 | 02 05 00 FF AF FE |
| SD2 | 2->8 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |
| SD2 | 2<-8 | DL | Get Diagnostics | Res | 62<-60 | 6 | 02 05 00 FF AF FE |
| SD2 | 2->10 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |
| SD2 | 2<-10 | DL | Get Diagnostics | Res | 62<-60 | 6 | 02 05 00 FF AF FE |
|  |  |  |  |  |  |  | B8 02 03 25 AF FE 00 E0 60 |
|  |  |  |  |  |  |  | 00 09 05 00 00 01 FF FF 00 |
| SD2 | 2->7 | SRD\_HIGH | Set Parameters | Req | 62->61 | 19 | 00 |
| ACK |  |  | Short acknowledge | Res |  |  |  |
|  |  |  |  |  |  |  | B8 02 03 25 AF FE 00 E0 60 |
|  |  |  |  |  |  |  | 00 09 05 00 00 01 FF FF 00 |
| SD2 | 2->8 | SRD\_HIGH | Set Parameters | Req | 62->61 | 19 | 00 |
| ACK |  |  | Short acknowledge | Res |  |  |  |
|  |  |  |  |  |  |  | B8 02 03 0B AF FE 00 C0 60 |
|  |  |  |  |  |  |  | 08 11 07 00 00 01 07 08 01 |
|  |  |  |  |  |  |  | 06 00 01 08 08 02 07 00 04 |
|  |  |  |  |  |  |  | 0C 81 00 00 05 00 00 01 FF |
| SD2 | 2->10 | SRD\_HIGH | Set Parameters | Req | 62->61 | 39 | FF 00 00 |
| ACK |  |  | Short acknowledge | Res |  |  |  |
|  |  |  |  |  |  |  | 42 00 00 01 42 00 00 02 82 |
|  |  |  |  |  |  |  | 00 00 03 C1 03 03 04 C1 01 |
| SD2 | 2->7 | SRD\_HIGH | Check Config | Req | 62->62 | 20 | 01 05 |
| ACK |  |  | Short acknowledge | Res |  |  |  |
|  |  |  |  |  |  |  | 42 00 00 01 42 00 00 02 82 |
|  |  |  |  |  |  |  | 00 00 03 C1 03 03 04 C1 01 |
| SD2 | 2->8 | SRD\_HIGH | Check Config | Req | 62->62 | 20 | 01 05 |
| ACK |  |  | Short acknowledge | Res |  |  |  |
|  |  |  |  |  |  |  | 42 00 00 01 42 00 00 02 82 |
|  |  |  |  |  |  |  | 00 00 03 C1 03 03 04 C1 01 |
|  |  |  |  |  |  |  | 01 05 42 00 FD 00 42 03 FD |
| SD2 | 2->10 | SRD\_HIGH | Check Config | Req | 62->62 | 36 | 03 03 00 00 FF 03 00 00 FF |
| ACK |  |  | Short acknowledge | Res |  |  |  |
| SD2 | 2->7 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |
| SD2 | 2<-7 | DL | Get Diagnostics | Res | 62<-60 | 6 | 02 0C 00 02 AF FE |
| SD2 | 2->8 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |
| SD2 | 2<-8 | DL | Get Diagnostics | Res | 62<-60 | 6 | 02 0C 00 02 AF FE |
| SD2 | 2->10 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |
| SD2 | 2<-10 | DL | Get Diagnostics | Res | 62<-60 | 6 | 02 0C 00 02 AF FE |
| SD2 | 2->7 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |
| SD2 | 2<-7 | DL | Get Diagnostics | Res | 62<-60 | 6 | 02 0C 00 02 AF FE |
| SD2 | 2->8 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |
| SD2 | 2<-8 | DL | Get Diagnostics | Res | 62<-60 | 6 | 02 0E 00 02 AF FE |
| SD2 | 2->10 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SD2 | 2<-10 | DL | Get Diagnostics | Res | 62<-60 | 6 | 02 0C 00 02 AF FE |
| SD2 | 2->7 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |
| SD2 | 2<-7 | DL | Get Diagnostics | Res | 62<-60 | 6 | 02 0E 00 02 AF FE |
| SD2 | 2->8 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |
| SD2 | 2<-8 | DL | Get Diagnostics | Res | 62<-60 | 6 | 00 0E 00 02 AF FE |
| SD2 | 2->10 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |
| SD2 | 2<-10 | DL | Get Diagnostics | Res | 62<-60 | 6 | 02 0E 00 02 AF FE |
| SD2 | 2->7 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |
| SD2 | 2<-7 | DL | Get Diagnostics | Res | 62<-60 | 6 | 00 0E 00 02 AF FE |
| SD2 | 2->8 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |
| SD2 | 2<-8 | DL | Get Diagnostics | Res | 62<-60 | 6 | 00 0C 00 02 AF FE |
| SD2 | 2->10 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |
| SD2 | 2<-10 | DL | Get Diagnostics | Res | 62<-60 | 6 | 00 0E 00 02 AF FE |
| SD1 | 2->8 |  |  | Req |  |  |  |
| SD2 | 127<-8 | DL | Data Exchange | Res |  | 8 | 08 00 00 00 00 00 00 00 |
| SD2 | 2->7 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |
| SD2 | 2<-7 | DL | Get Diagnostics | Res | 62<-60 | 6 | 00 0C 00 02 AF FE |
| SD2 | 2->10 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |
| SD2 | 2<-10 | DL | Get Diagnostics | Res | 62<-60 | 6 | 00 0E 00 02 AF FE |
| SD1 | 2->7 |  |  | Req |  |  |  |
| SD2 | 127<-7 | DL | Data Exchange | Res |  | 8 | 07 E0 00 00 00 00 00 00 |
| SD1 | 2->8 |  |  | Req |  |  |  |
| SD2 | 127<-8 | DL | Data Exchange | Res |  | 8 | 08 00 00 00 00 00 00 00 |
| SD2 | 2->10 | SRD\_HIGH | Get Diagnostics | Req | 62->60 | 0 |  |
| SD2 | 2<-10 | DL | Get Diagnostics | Res | 62<-60 | 6 | 00 0C 00 02 AF FE |
| SD1 | 2->7 |  |  | Req |  |  |  |
| SD2 | 127<-7 | DL | Data Exchange | Res |  | 8 | 07 E0 00 00 00 00 00 00 |
| SD1 | 2->8 |  |  | Req |  |  |  |
| SD2 | 127<-8 | DL | Data Exchange | Res |  | 8 | 08 00 00 00 00 00 00 00 |
| SD1 | 2->10 | SRD\_HIGH | Data Exchange | Req |  |  |  |
| SD2 | 2<-10 | DL | Data Exchange | Res |  | 13 | 0A 80 00 00 00 00 00 00 00  00 00 00 00 |
| SD1 | 2->7 |  |  | Req |  |  |  |
| SD2 | 127<-7 | DL | Data Exchange | Res |  | 8 | 07 E0 00 00 00 00 00 00 |
| SD1 | 2->8 |  |  | Req |  |  |  |
| SD2 | 127<-8 | DL | Data Exchange | Res |  | 8 | 08 00 00 00 00 00 00 00 |
| SD1 | 2->10 | SRD\_HIGH | Data Exchange | Req |  |  |  |
| SD2 | 2<-10 | DL | Data Exchange | Res |  | 13 | 0A 80 00 00 00 00 00 00 00  00 00 00 00 |
| SD1 | 2->7 |  |  | Req |  |  |  |
| SD2 | 127<-7 | DL | Data Exchange | Res |  | 8 | 07 E0 00 00 00 00 00 00 |
| SD1 | 2->8 |  |  | Req |  |  |  |
| SD2 | 127<-8 | DL | Data Exchange | Res |  | 8 | 08 00 00 00 00 00 00 00 |
| SD1 | 2->10 | SRD\_HIGH | Data Exchange | Req |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SD2 | 2<-10 | DL | Data Exchange | Res |  | 13 | 0A 80 00 00 00 00 00 00 00  00 00 00 00 |
| SD1 | 2->7 |  |  | Req |  |  |  |
| SD2 | 127<-7 | DL | Data Exchange | Res |  | 8 | 07 E0 00 00 00 00 00 00 |
| SD1 | 2->8 |  |  | Req |  |  |  |
| SD2 | 127<-8 | DL | Data Exchange | Res |  | 8 | 08 00 00 00 00 00 00 00 |
| SD1 | 2->10 | SRD\_HIGH | Data Exchange | Req |  |  |  |
| SD2 | 2<-10 | DL | Data Exchange | Res |  | 13 | 0A 80 00 00 00 00 00 00 00  00 00 00 00 |
| SD1 | 2->7 |  |  | Req |  |  |  |
| SD2 | 127<-7 | DL | Data Exchange | Res |  | 8 | 07 E0 00 00 00 00 00 00 |
| SD1 | 2->8 |  |  | Req |  |  |  |
| SD2 | 127<-8 | DL | Data Exchange | Res |  | 8 | 08 00 00 00 00 00 00 00 |
| SD1 | 2->10 | SRD\_HIGH | Data Exchange | Req |  |  |  |
| SD2 | 2<-10 | DL | Data Exchange | Res |  | 13 | 0A 80 00 00 00 00 00 00 00  00 00 00 00 |
| SD1 | 2->7 |  |  | Req |  |  |  |
| SD2 | 127<-7 | DL | Data Exchange | Res |  | 8 | 07 E0 00 00 00 00 00 00 |
| SD1 | 2->8 |  |  | Req |  |  |  |
| SD2 | 127<-8 | DL | Data Exchange | Res |  | 8 | 08 00 00 00 00 00 00 00 |
| SD1 | 2->10 | SRD\_HIGH | Data Exchange | Req |  |  |  |
| SD2 | 2<-10 | DL | Data Exchange | Res |  | 13 | 0A 80 00 00 00 00 00 00 00  00 00 00 00 |
| SD1 | 2->7 |  |  | Req |  |  |  |
| SD2 | 127<-7 | DL | Data Exchange | Res |  | 8 | 07 E0 00 00 00 00 00 00 |
| SD1 | 2->8 |  |  | Req |  |  |  |
| SD2 | 127<-8 | DL | Data Exchange | Res |  | 8 | 08 00 00 00 00 00 00 00 |

**Figure 3-27 : Bus monitoring**

## Notes:

## Interrupt Indication Function

The VPC3+ generates indications based on internals events. The indications can observed by means of polling or interrupt.

The user can mask each interrupt by setting the corresponding bit in the Interrupt Mask Register (DpCfg.h, ). If interrupt are masked, the application must poll the Interrupt Request Register for active indications.

For interrupt handling and poll-mode refer to file “dp\_isr.c”.

### Reading the Indication

The user receives the event which has caused the interrupt by reading the Interrupt Register:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Address** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 02H  (Intel) | DXB\_Out | New\_Ext\_Prm | DXB\_Link\_Error | User\_Timer\_ Clock | WD\_DP\_ Mode\_Timeout | Baud\_Rate\_ Detect | Go/Leave Data\_EX | MAC\_Reset \ Clock\_Sync | Interrupt Register 7 .. 0 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Address** | **Bit Position** | | | | | | | | **Designation** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| 03H  (Intel) | FDL\_ind | Poll\_End\_ind | DX\_Out | Diag\_Buffer\_  Changed | New\_Prm\_  Data | New\_Cfg\_  Data | New\_SSA\_  Data | New\_GC  Command | Interrupt-Register 15 .. 8 |

|  |  |
| --- | --- |
| **Indication** | **Description** |
| VPC3\_GET\_IND\_MAC\_RESET | After processing the current request, the VPC3+ has entered the offline state (by setting the ‘Go\_Offline’ bit). |
| VPC3\_GET\_IND\_CLOCK\_SYNC | The VPC3+ has received a Clock-Sync- telegram. |
| VPC3\_ GET\_IND \_GO\_LEAVE\_DATA\_EX | The DP\_SM has entered the ‘DATA\_EX’ state or has exited it. |
| VPC3\_ GET\_IND \_BAUDRATE\_DETECT | The VPC3+ has left the ‘Baud\_Search state’ and has found a baud rate. |
| VPC3\_ GET\_IND \_DP\_WD\_TIMEOUT | In the ‘DP\_Control’ WD state , the watchdog timer has expired. |
| VPC3\_ GET\_IND \_USER\_TIMER\_CLOCK | The time base of the User\_Timer\_Clock has expired (1/10ms). |
| VPC3\_GET\_IND\_DXB\_LINK\_ERROR | The VPC3+ has updated the DXB Link structure. The data is available in the DXB\_Link\_Table buffer. |
| VPC3\_GET\_IND\_NEW\_EXT\_PRM\_DATA | The VPC3+ has received ‘Set\_Ext\_Param Message’ and has made the data available in the Prm buffer. |
| VPC3\_GET\_IND\_DXB\_OUT | The VPC3+ has received new data from the DXB Publisher. The data is available in the DXB\_OUT buffer. |
| VPC3\_GET\_IND\_NEW\_GC\_COMMAND | The VPC3+ has received a ‘Global\_Control Message’ with a changed ‘GC\_Command Byte’ and has stored this byte in the ‘R\_GC\_Command’ RAM cell. |
| VPC3\_ GET\_IND \_NEW\_SSA\_DATA | The VPC3+ has received ‘Set\_Slave\_Address Message’ and has made the data available in the SSA buffer. |
| VPC3\_ GET\_IND \_NEW\_CFG\_DATA | The VPC3+ has received Check\_Cfg Message’ and has made the data available in the Cfg buffer. |
| VPC3\_ GET\_IND \_NEW\_PRM\_DATA | The VPC3+ has received ‘Set\_Param Message’ and has made the data available in the Prm buffer. |
| VPC3\_GET\_IND\_DIAG\_BUF\_CHANGED | Requested by ‘New\_Diag\_Cmd’ , the VPC3+ has Exchanged the diagnostics buffer and has made the old buffer available again to the user. |
| VPC3\_ GET\_IND \_DX\_OUT | The VPC3+ has received a ‘Write\_Read\_Data Message’ and has made the new output data available in the N buffer. For ‘Power\_On’ and for ‘Leave\_Master’, the VPC3+ clears the N buffer contents and also generates this interrupt. |
| VPC3\_GET\_IND\_POLL\_END\_IND | The master has fetched the FDL response. |
| VPC3\_GET\_IND\_FDL\_IND | The VPC3+ has received a FDL indication. |

**Figure 4-1 : Interrupt indication**

### Acknowledging the Indication

The user acknowledges the indication received through the interrupt routine by writing to the Interrupt Acknowledge Register:

VPC3\_CON\_IND\_MAC\_RESET() VPC3\_CON\_IND\_CLOCK\_SYNC() VPC3\_CON\_IND\_GO\_LEAVE\_DATA\_EX() VPC3\_CON\_IND\_BAUDRATE\_DETECT() VPC3\_CON\_IND\_DP\_WD\_TIMEOUT() VPC3\_CON\_IND\_USER\_TIMER\_CLOCK() VPC3\_CON\_IND\_DXB\_LINK\_ERROR() VPC3\_CON\_IND\_NEW\_EXT\_PRM\_DATA() VPC3\_CON\_IND\_DXB\_OUT() VPC3\_CON\_IND\_NEW\_GC\_COMMAND() VPC3\_CON\_IND\_NEW\_SSA\_DATA() VPC3\_CON\_IND\_DIAG\_BUF\_CHANGED() VPC3\_CON\_IND\_DX\_OUT() VPC3\_CON\_IND\_POLL\_END\_IND() VPC3\_CON\_IND\_FDL\_IND()

**Interrupt 10 (New\_Cfg\_Data) and interrupt 11 (New\_Prm\_Data) can not be acknowledged with the Interrupt Acknowledge Register. They are acknowledged by reading from**

VPC3\_SET\_PRM\_DATA\_OK() VPC3\_SET\_PRM\_DATA\_NOK()

VPC3\_SET\_CFG\_DATA\_OK() VPC3\_SET\_CFG\_DATA\_NOK()

### Ending the Indication

The EOI-bit (End Of Interrupt) in mode register 1, bit 1, ends the indication sequence / interrupt function:

|  |  |  |
| --- | --- | --- |
| **VPC3\_SET\_EOI()** | | |
| Function | Ends indication of interrupt function | |
| Parameter | None | |
| Return Value | None |  |

**Figure 4-2 : Function VPC3\_SET\_EOI()**

### Polling the Indication

The user can poll indications via the Interrupt Request Register:

VPC3\_POLL\_IND\_MAC\_RESET() VPC3\_POLL\_IND\_CLOCK\_SYNC() VPC3\_POLL\_IND\_GO\_LEAVE\_DATA\_EX() VPC3\_POLL\_IND\_BAUDRATE\_DETECT() VPC3\_POLL\_IND\_DP\_WD\_TIMEOUT() VPC3\_POLL\_IND\_USER\_TIMER\_CLOCK() VPC3\_POLL\_IND\_DXB\_LINK\_ERROR() VPC3\_POLL\_IND\_NEW\_EXT\_PRM\_DATA() VPC3\_POLL\_IND\_DXB\_OUT() VPC3\_POLL\_IND\_NEW\_GC\_COMMAND() VPC3\_POLL\_IND\_NEW\_SSA\_DATA() VPC3\_POLL\_IND\_DIAG\_BUF\_CHANGED() VPC3\_POLL\_IND\_DX\_OUT() VPC3\_POLL\_IND\_POLL\_END\_IND() VPC3\_POLL\_IND\_FDL\_IND()

Poll indications can be acknowledged via the Interrupt Acknowledge Register:

VPC3\_CON\_IND\_MAC\_RESET() VPC3\_CON\_IND\_CLOCK\_SYNC() VPC3\_CON\_IND\_GO\_LEAVE\_DATA\_EX() VPC3\_CON\_IND\_BAUDRATE\_DETECT() VPC3\_CON\_IND\_DP\_WD\_TIMEOUT() VPC3\_CON\_IND\_USER\_TIMER\_CLOCK() VPC3\_CON\_IND\_DXB\_LINK\_ERROR() VPC3\_CON\_IND\_NEW\_EXT\_PRM\_DATA() VPC3\_CON\_IND\_DXB\_OUT() VPC3\_CON\_IND\_NEW\_GC\_COMMAND() VPC3\_CON\_IND\_NEW\_SSA\_DATA() VPC3\_CON\_IND\_DIAG\_BUF\_CHANGED() VPC3\_CON\_IND\_DX\_OUT() VPC3\_CON\_IND\_POLL\_END\_IND() VPC3\_CON\_IND\_FDL\_IND()

## Parameter Data

### Checking the Parameter Data

Checking of parameter data is application dependent. Therefore the user is responsible for checking the received user specific parameter data. With the interrupt VPC3\_GET\_IND\_NEW\_PRM\_DATA the function VPC3\_Isr is called and then, if necessary, the user specific parameter data checking sequence within the interrupt routine.

#### Callback function:

|  |  |  |
| --- | --- | --- |
| **DP\_ERROR\_CODE DpPrm\_ChkNewPrmData( MEM\_UNSIGNED8\_PTR pbPrmData, uint8\_t bPrmLength )** | | |
| Function | Checking parameter data | |
| Parameter | pbPrmData | Pointer to parameter data |
| bPrmLength | Length of parameter data |
| Return Value | DP\_OK DP\_NOK | Parameter data OK  **\*Error Parameter data not OK** |

**Figure 4-3 : Function DpPrm\_ChkNewPrmData()**

#### Functions:

|  |  |  |
| --- | --- | --- |
| **uint8\_t VPC3\_GET\_PRM\_LEN()** | | |
| Function | Get the length of the received parameter data | |
| Parameter | None | |
| Return Value | Length of prm data |  |

**Figure 4-4 : Function VPC3\_GET\_PRM\_LEN**

|  |  |  |
| --- | --- | --- |
| **VPC3\_UNSIGNED8\_PTR VPC3\_GET\_PRM\_BUF\_PTR ()** | | |
| Function | Fetch buffer pointer of the parameter buffer. | |
| Parameter | None | |
| Return Value | pointer to the parameter data buffer |  |

**Figure 4-5 : Function VPC3\_GET\_PRM\_BUF\_PTR**

|  |  |  |
| --- | --- | --- |
| **uint8\_t VPC3\_SET\_PRM\_DATA\_OK()** | | |
| Function | Positive acknowledge of the checked parameter data. | |
| Parameter | None | |
| Return Value | VPC3\_PRM\_FINISHED | No further parameter assignment message is present => end of sequence. |
| VPC3\_PRM\_CONFLICT | Another parameter assignment message is present! => repeat check of requested parameter assignment. |
| VPC3\_PRM\_NOT\_ALLOWED | Access in present bus mode is not permitted. For example, it is possible that the watchdog has expired during verification. |

**Figure 4-6 : Function VPC3\_SET\_PRM\_DATA\_OK**

|  |  |  |
| --- | --- | --- |
| **uint8\_t VPC3\_SET\_PRM\_DATA\_NOK()** | | |
| Function | Negative acknowledge of the checked parameter data. | |
| Parameter | None | |
| Return Value | VPC3\_PRM\_FINISHED | No further parameter assignment message is present => end of sequence. |
| VPC3\_PRM\_CONFLICT | Another parameter assignment message is present! => repeat check of requested parameter assignment. |
| VPC3\_PRM\_NOT\_ALLOWED | Access in present bus mode is not permitted. For example, it is possible that the watchdog has expired during verification. Verifying the parameter data (and possibly series-connected functions in the application) are to be cancelled. |

**Figure 4-7 : Function VPC3\_SET\_PRM\_DATA\_NOK()**

Acknowledging the New\_Prm\_Data interrupt by using one of these com- mands means, that the corresponding interrupt request bit is cleared. The New\_Prm\_Data interrupt can not be acknowledged via the Interrupt Acknowledge Register

#### Caution:

When both, configuration settings and parameter settings, are received, it is mandatory to verify and acknowledge parameter data first. Then the configuration settings may be processed.

### Parameter Data Structure

VPC3+ evaluates the first seven data bytes (without user prm data), or the first eight data bytes (with user prm data). The first seven bytes are specified according to the standard. The next three bytes are used for the extended profibus services DPV1 and DPV2. The additional bytes are available to the application.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 0 | Lock\_ Req | Unlock\_ Req | Sync\_ Req | Freeze\_ Req | WD\_On | Reserved | Reserved | Reserved | Station Status |
| 1 |  |  |  |  |  |  |  |  | WD\_Fact\_1 |
| 2 |  |  |  |  |  |  |  |  | WD\_Fact\_2 |
| 3 |  |  |  |  |  |  |  |  | MinTSDR |
| 4 |  |  |  |  |  |  |  |  | Ident\_Number\_High |
| 5 |  |  |  |  |  |  |  |  | Ident\_Number\_Low |
| 6 |  |  |  |  |  |  |  |  | Group\_Ident |
| 7 |  |  |  |  |  |  |  |  |  |
| : | DPV1\_STATUS1..3 |
| 9 |  |
| 10 |  |  |  |  |  |  |  |  |  |
| : | User\_Prm\_Data |
| 243 |  |

**Figure 4-8 : Format of the Set\_Param Telegram**

#### Don’t use DPV1\_STATUS1..3 as User\_Prm\_Data.

|  |  |
| --- | --- |
|  | **DPV1\_STATUS1:** |
| Bit 7 | **DPV1\_Enable:**  0 = The slave is operated in the DP mode. (default state)  1 = The slave is operated in the DPV1 mode. |
| Bit 6 | **\*Fail\_Safe:**  0 = The slave is not operated in the Fail Safe mode (default state).  1 = The slave is operated in the Fail Safe mode. |
| Bit 5 | **Publisher\_Enable:**  0 = The slave is not operated in the DXB Publisher mode (default state).  1 = The slave is operated in the DXB Publisher mode. |
| Bit 4-3 | **Reserved:** To be parameterized with ‘0’ |
| Bit 2 | **WD\_Base:** Watchdog Time Base  0 = Watchdog time base is 10 ms (default state)  1 = Watchdog time base is 1 ms |
| Bit 1 | **Dis\_Stop\_Control:** Disable Stop-Bit Control  0 = Stop-bit monitoring in the receiver is enabled (default state)  1 = Stop-bit monitoring in the receiver is disabled |
| Bit 0 | **Dis\_Start\_Control:** Disable Start-Bit Control  0 = Start-bit monitoring in the receiver is enabled (default state)  1 = Start-bit monitoring in the receiver is disabled |

**Figure 4-9 : DPV1\_STATUS1**

\*)If the DP-Slave requires the Fail Safe mode and the master does not set this bit, the slave has to reject the parameter assignment.

|  |  |
| --- | --- |
|  | **DPV1\_STATUS2:** |
| Bit 7 | **Enable\_Pull\_Plug\_Alarm:**  0 = Enable\_Pull\_Plug\_Alarm disabled  1 = Enable\_Pull\_Plug\_Alarm enabled. |
| Bit 6 | **Enable\_Process\_Alarm:**  0 = Enable\_Process\_Alarm disabled  1 = Enable\_Process\_Alarm enabled. |
| Bit 5 | **Enable\_Diagnostic\_Alarm:**  0 = Enable\_Diagnostic\_Alarm disabled  1 = Enable\_Diagnostic\_Alarm enabled. |
| Bit 4 | **Enable\_Manufacturer\_Specific\_Alarm:**  0 = Enable\_Manufacturer\_Specific\_Alarm disabled  1 = Enable\_Manufacturer\_Specific\_Alarm enabled. |
| Bit 3 | **Enable\_Status\_Alarm:**  0 = Enable\_Status\_Alarm disabled  1 = Enable\_Status\_Alarm enabled. |
| Bit 2 | **Enable\_Update\_Alarm:**  0 = Enable\_Update\_Alarm disabled  1 = Enable\_Update\_Alarm enabled. |
| Bit 1 | **Reserved:** To be parameterized with ‘0’ |
| Bit 0 | **Chk\_Cfg\_Mode:**  0 = Chk\_Cfg according to EN50170 (default state)  1 = User-specific evaluation of Chk\_Cfg |

**Figure 4-10 : DPV1\_STATUS2**

|  |  |
| --- | --- |
|  | **DPV1\_STATUS3:** |
| bit 7-5 | **Reserved:** To be parameterized with ‘0’ |
| bit 4 | **IsoM\_Req:** Isochron Mode Request  0 = Isochron Mode disabled  1 = Isochron Mode enabled |
| bit 3 | **Prm\_Structure:**  0 = Prm telegram according to EN50170  1 = Prm telegram in structured form (DPV2 extension) |
| bit 0-2 | **Alarm\_Mode:** limits the number of active alarms  0 = 1 alarm of each type  1 = 2 alarms in total  2 = 4 alarms in total  3 = 8 alarms in total  4 = 12 alarms in total  5 = 16 alarms in total  6 = 24 alarms in total  7 = 32 alarms in total |

**Figure 4-11 : DPV1\_STATUS3**

If **Prm\_Structure set to 1**, the prm-data are in the structured form:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 0 |  |  |  |  |  |  |  |  |  |
| : | See above |
| 6 |  |
| 7 |  |  |  |  |  |  |  |  | DPV1\_STATUS1 |
| 8 |  |  |  |  |  |  |  |  | DPV1\_STATUS2 |
| 9 |  |  |  |  | 1 |  |  |  | DPV1\_STATUS3 |
| 10 |  |  |  |  |  |  |  |  | Structured\_Length |
| 11 |  |  |  |  |  |  |  |  | Structure\_Type 0x02: PrmCmd  0x03: DXB LinkTable 0x04: ISOCHRON  0x07: DXB Subscriber 0x08: Time AR  0x81: USER\_PRM |
| 12 |  |  |  |  |  |  |  |  | Slotnumber |
| 13 |  |  |  |  |  |  |  |  | Reserved |
| 14 |  |  |  |  |  |  |  |  |  |
| : | Data |
| : |  |

**Figure 4-12 : Structured Format of the Set\_Param Telegram**

## Configuration Data

### Checking Configuration Data

Checking of configuration data is application dependent. Therefore the user is responsible for checking the received configuration data. With the interrupt VPC3\_INT\_NEW\_CFG\_DATA function VPC3\_Isr is called and then, if necessary, the user specific configuration data checking sequence within the interrupt routine.

#### Callback function:

|  |  |  |
| --- | --- | --- |
| **E\_DP\_CFG\_ERROR DpCfg\_ChkNewCfgData( MEM\_UNSIGNED8\_PTR pbCfgData, uint8\_t bCfgLength )** | | |
| Function | Checking configuration data | |
| Parameter | pbCfgData | Pointer to configuration data |
| bCfgLength | Length of configuration data |
| Return Value | DP\_CFG\_OK DP\_CFG\_FAULT DP\_CFG\_UPDATE | Configuration data OK  **\*Error Configuration data not OK**  Configuration data is OK, but different from ReadCfg-buffer. |

**Figure 4-13 : Function DpCfg\_ChkNewCfgData()**

#### Functions:

|  |  |  |
| --- | --- | --- |
| **uint8\_t VPC3\_GET\_READ\_CFG\_LEN()**  **uint8\_t VPC3\_GET\_CFG\_LEN()** | | |
| Function | Get the length of the configuration data. | |
| Parameter | None | |
| Return Value | Length of cfg data |  |

**Figure 4-14 : Function VPC3\_GET\_CFG\_LEN**

|  |  |  |
| --- | --- | --- |
| **VPC3\_UNSIGNED8\_PTR VPC3\_GET\_READ\_CFG\_BUF\_PTR ()**  **VPC3\_UNSIGNED8\_PTR VPC3\_GET\_CFG\_BUF\_PTR ()** | | |
| Function | Fetch buffer pointer of the configuration buffer. | |
| Parameter | None | |
| Return Value | pointer to the configuration data buffer |  |

**Figure 4-15 : Function VPC3\_GET\_CFG\_BUF\_PTR**

Within the verification function, the user compares the received Cfg\_Data with the Real\_Cfg\_Data (Real\_Cfg\_Data was set during initialization).

|  |  |  |
| --- | --- | --- |
| **uint8\_t VPC3\_SET\_CFG\_DATA\_OK()** | | |
| Function | Positive acknowledge of the checked configuration data. | |
| Parameter | None | |
| Return Value | VPC3\_CFG\_FINISHED | No further configuration message is present  => end of sequence. |
| VPC3\_CFG\_CONFLICT | An additional configuration message is present! => Repeat verification of the requested configuration. |
| VPC3\_CFG\_NOT\_ALLOWED | Access is not permitted in the present bus mode. For example, it is possible the watchdog has run out during verification. The verification of the configuration data (and possibly subsequent functions in the application) are to be cancelled. |

**Figure 4-16 : Function VPC3\_SET\_CFG\_DATA\_OK**

|  |  |  |
| --- | --- | --- |
| **uint8\_t VPC3\_SET\_CFG\_DATA\_NOK()** | | |
| Function | Negative acknowledge of the checked configuration data. | |
| Parameter | None | |
| Return Value | VPC3\_CFG\_FINISHED | No further configuration message is present  => end of sequence. |
| VPC3\_CFG\_CONFLICT | An additional configuration message is present! => Repeat verification of the requested configuration. |
| VPC3\_CFG\_NOT\_ALLOWED | Access is not permitted in the present bus mode. For example, it is possible the watchdog has run out during verification. The verification of the configuration data (and possibly subsequent functions in the application) are to be cancelled. |

**Figure 4-17 : Function VPC3\_SET\_CFG\_DATA\_NOK**

Acknowledging the New\_Cfg\_Data interrupt by using one of these com- mands means, that the corresponding interrupt request bit is cleared. The New\_Cfg\_Data interrupt can not be acknowledged via the Interrupt Acknowledge Register

#### Caution:

When both, configuration settings and parameter settings, are received, it is mandatory to verify and acknowledge parameter data first. Then the configuration settings may be processed.

### Configuration Data Formats

#### General format:

Bit-No

**data length**

7 6 5 4 3 2 1 0

**in- / output** 00 = special identifier format 01 = input

10 = output

11 = input-output

**length structure** 0 = byte

1 = word

**consistency across** 0 = byte or word

1 = total length

For example, the identifiers correspond to 14 hex = 5 bytes input

27 hex = 8 bytes output

**Figure 4-18 : General Configuration Data Format**

In order to cover complexer configurations, greater flexibility is attained in the case of PROFIBUS DP through a special expansion of the actual identification system. In addition, this special ID format makes it possible to determine the number of the input- and output bytes of this ID. Furthermore, user-specific data can be added.

#### Special format:

Bit-No

0

1

2

3

4

5

6

7

**length of manufacturer data fixed to 00**

**in- / output** 00 = free place

01 = it follows 1 length byte for inputs

10 = it follows 1 length byte for outputs

11 = it follows 1 length byte for outputs

1 length byte for inputs

**Figure 4-19 : Special Configuration Data Format**

The length indication for manufacturer-specific data is to be interpreted as follows:

|  |  |
| --- | --- |
| 0 | No manufacturer-specific data follows; it is not to be present in the Real\_Cfg\_Data. |
| 1 to 14 | Manufacturer-specific data of the specified length follows; it has to agree with the data contained in Real\_Cfg\_Data |
| 15 | No manufacturer-specific data follows; there is no check. |

The structure of the length bytes looks like this:

Bit-No

0

1

2

3

4

5

6

7

**length of inputs / outputs**

**length structure**

**consistency over**

0 = byte

1 = word

0 = byte or word

1 = whole length

**Figure 4-20 : Special Configuration Data Format**

For example: C0hex, 87hex,84hex (8 bytes output, 5 bytes input)

## Transfer of Output Data

VPC3\_INT\_DX\_OUT in the interrupt function VPC3\_Isr() indicates the receipt of output data from the DP-Master. The function VPC3\_GetDoutBufPtr () returns the buffer pointer, and also the state of the Dout-buffer. The lengths of the outputs are not transferred with every update. The length agrees with the length transferred with VPC3\_SetIoDataLength(), otherwise VPC3+ would branch to the WAIT\_PRM state.

|  |  |  |
| --- | --- | --- |
| **VPC3\_UNSIGNED8\_PTR VPC3\_GetDoutBufPtr ( MEM\_UNSIGNED8\_PTR pbState )** | | |
| Function | Fetch buffer pointer and state of the output buffer. | |
| Parameter | Pointer to variable into which the state of the output buffer is to be written | |
| Return Value | pointer to the output data buffer  NIL, if no diagnostics buffer in the ‘U’ state |  |
| state of the output buffer | NEW\_DOUT\_BUF DOUT\_BUF\_CLEARED |

**Figure 4-21 : Function VPC3\_Get\_DoutBufPtr()**

**The input-/output data length can be reconfigured with the functions described in the Initialization section (VPC3\_SetIoDataLength(), VPC3\_CalculateInpOutpLength(),...).**

## Transfer of Input Data

As described, the application has to fetch a buffer for the input data with the VPC3\_GetDinBufPtr() function before the first entry of its input data. With the command

|  |  |  |
| --- | --- | --- |
| **uint8\_t VPC3\_INPUT\_UPDATE ()** | | |
| Function | Change the input buffer. | |
| Parameter | None | |
| Return Value | New U-buffer | 1 = Din\_Buf\_Ptr1  2 = Din\_Buf\_Ptr2  3 = Din\_Buf\_Ptr3 |

**Figure 4-22 : Function VPC3\_INPUT\_UPDATE**

the user can repeatedly transfer the current input data from the user to the VPC3+. The length of the inputs is not transferred with every update. The length must agree with the length transferred with function VPC3\_SetIoDataLength().

|  |  |  |
| --- | --- | --- |
| **VPC3\_UNSIGNED8\_PTR VPC3\_GetDinBufPtr ()** | | |
| Function | Fetch buffer pointer of the input buffer. | |
| Parameter | None | |
| Return Value | pointer to the input data buffer  NIL, if no diagnostics buffer in the ‘U’ state |  |

**Figure 4-23 : Function VPC3\_GetDinBufPtr**

**The input-/output data length can be reconfigured with the functions described in the Initialization section (VPC3\_SetIoDataLength(), VPC3\_CalculateInpOutpLength(),...).**

## Diagnostic

### Transferring Diagnostic Data

With the function VPC3\_SetDiagnosis(), the user can transfer diagnostic data to the VPC3+. Before calling this function, the user has to get a pointer to the free diagnosis buffer with the function VPC3\_GetDiagBufPtr.

|  |  |  |
| --- | --- | --- |
| **DP\_ERROR\_CODE VPC3\_SetDiagnosis( MEM\_UNSIGNED8\_PTR pbToUserDiagData, uint8\_t bUserDiagLength, uint8\_t bDiagControl, uint8\_t bCheckDiagFlag )** | | |
| Function | Transfer user diagnostic to VPC3+ | |
| Parameter | pbToUserDiagData | Pointer to user diagnostic block |
| bUserDiagLength | Length of user diagnostic block |
| bDiagControl | 0: reset Diag.ExtDiag / Diag.StatDiag-bit 1: set Diag.ExtDiag-bit  2: set Diag.StatDiag-bit |
| bCheckDiagFlag | FALSE: don’t check DIAG\_FLAG of VPC3+ TRUE: check DIAG\_FLAG of VPC3+  (see VPC3+ Status register) |
| Return value | DP\_OK | Execution OK, message copied to VPC3+ |
| OLD\_DIAG\_NOT\_SEND\_ERROR | Error, wait because last diagnostic message isn’t send |
| BUFFER\_LENGTH\_ERROR | Error, diagnostic message is too long |
| NO\_BUFFER\_ERROR | Error, wait no VPC3+ diagnostic buffer available |
| CONTROL\_BYTE\_ERROR | Error of bDiagControl |
| BUFFER\_ERROR | Error, wrong diagnostic header |
| NOT\_POSSIBLE\_ERROR | Error, unknown error |

**Figure 4-24 : Function VPC3\_SetDiagnosis()**

|  |  |  |
| --- | --- | --- |
| **VPC3\_UNSIGNED8\_PTR VPC3\_GetDiagBufPtr ()** | | |
| Function | Fetch buffer pointer of the diagnostic buffer. | |
| Parameter | None | |
| Return Value | Pointer to the diagnostics buffer  NIL, if no diagnostics buffer in the ‘U’ state |  |

**Figure 4-25 : Function VPC3\_GetDiagBufPtr**

### Structure of diagnostic block

Structure of the data block to be transferred for expanded diagnostics:

|  |  |  |
| --- | --- | --- |
| **Byte** | **Diagnosis Data** | **Comment** |
| 0 | Station Status\_1 | Byte 0 to 5 permanent diagnostic header |
| 1 | Station Status\_2 |
| 2 | Station Status\_3 |
| 3 | Diag.Master\_Add |
| 4 | Ident\_Number\_High |
| 5 | Ident\_Number\_Low |
| 6  : 243 | Ext\_Diag\_Data | Start of user diagnostic in the DP Standard format |

**Figure 4-26 : Structure of diagnosticv block**

Station Status\_1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit |  | | | | | | | | | | | | | | |
| 7 | 6 |  | 5 |  | 4 |  | 3 |  |  | 2 | 1 |  | 0 |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **Diagnostic station does not exist (set by master)** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **Diag.station\_not\_ready** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **Slave not ready for DataExchange**  **Diag.Cfg\_Fault** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **Configuration Data do not match** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **Diag.Ext\_Diag** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **Slave has external diagnosis** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **Diag.Not\_Supported** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **Slave does not support the requested function**  **Diag.Invalid\_Slave\_Response** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **(set Slave to 0 permanently)** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **Diag.Prm\_Fault** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **bad parameters (Ident No. etc.)** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **Diag.Master\_Lock (set by master)** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **Slave was configured by other master** |

**Figure 4-27 : Structure of Station\_Status\_1**

Station Status\_2

bit

0

1

2

3

4

5

6

7

**Diag.Prm\_Req**

**Slave requires new configuration Diag.Stat\_Diag**

**statistic diagnosis**

**permanently at 1**

**Diag.WD\_ON Watchdog active**

**Diag.Freeze\_Mode**

**Freeze command was received**

**Sync\_Mode**

**Sync command was received reserved**

**Diag.Deactivated (set by master)**

**Figure 4-28 : Structure of Station\_Status\_2**

Station Status\_3

bit

0

1

2

3

4

5

6

7

**reserved Diag.Ext\_Overflow**

**Figure 4-29 : Structure of Station\_Status\_3**

### User specific diagnostic

The user-specific diagnostic can be filed in three different formats:

### Device related diagnostic

The diagnostic information can be coded as required:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Bit7** | **Bit6** | **Bit5-0** |
| Header byte | **0** | **0** | Block length in bytes,including header |
| Diagnostics field  ... | Coding of diagnostic is device specific,  can be specified as required | | |

**Figure 4-30 : Device related diagnostic**

### Identifier related diagnostic

For each used identifier byte at the configuration one bit is reserved. It is padded to byte limits. The bits which are not configured shall be set to zero. A set bit means that in this I/O area diagnostic is pending.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Bit7** | **Bit6** | **Bit5-0** | | | | | |
| Header byte | **0** | **1** | Block length in bytes,including header | | | | | |
| Bit structure  ... | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |

**Figure 4-31 : Device related diagnostic**

### Channel related diagnostic

In this block the diagnosed channels and the diagnostic reason are entered in turn. The length per entry is 3 octets.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Bit7** | **Bit6** | **Bit5** | **Bit4-0** |
| Header byte | **1** | **0** | Identification number | |
| Channel Number | Coding  Input/Output | | Channel number (0..63) | |
| Type of diagnosis | Coding  Channel type | | | Coding  Error type |

**Figure 4-32 : Channel related diagnostic**

|  |  |
| --- | --- |
| **Coding Input/Output** | |
| 00 | Reserved |
| 01 | Input |
| 10 | Output |
| 11 | Input / Output |

**Figure 4-33 : Coding Input/Output**

|  |  |
| --- | --- |
| **Coding Channel type** | |
| 000 | Reserved |
| 001 | Bit |
| 010 | 2 bit |
| 011 | 4 bit |
| 100 | Byte |
| 101 | Word |
| 110 | 2 words |
| 111 | Reserved |

**Figure 4-34 : Coding Channel type**

|  |  |
| --- | --- |
| **Coding Error type** | |
| 0 | reserved |
| 1 | short circuit |
| 2 | undervoltage |
| 3 | overvoltage |
| 4 | overload |
| 5 | overtemperature |
| 6 | line break |
| 7 | upper limit value exceeded |
| 8 | lower limit value exceeded |
| 9 | error |
| 10 | reserved |
| ... | ... |
| 15 | reserved |
| 16 | manufacturer specific |
| ... | ... |
| 31 | manufacturer specific |

**Figure 4-35 : Coding Error type**

Example: Structure of a diagnostic according to the pattern above:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MSB LSB** | | | | | | | |  |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |
| **0** | **0** | 0 | 0 | 0 | 1 | 0 | 0 | **Device** related diagnostic |
| Device specific | | | | | | | | Meaning of the bits |
| diagnostics field of | | | | | | | | is specified |
| length 3 | | | | | | | | manufacturer specific |
| **0** | **1** | 0 | 0 | 0 | 1 | 0 | 1 | **Identifier** related diagnostic |
|  |  |  |  |  |  |  | 1 | Identification number 0 has diagnostic |
|  |  |  | 1 |  |  |  |  | Identification number 12 has diagnostic |
|  |  |  |  |  |  | 1 |  | Identification number 17 has diagnostic |
|  |  |  |  |  |  |  |  |  |
| **1** | **0** | 0 | 0 | 0 | 0 | 0 | 0 | **Channel** related diagnostic, number 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | Channel 2 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | Overload, channel organized bit by bit |
| **1** | **0** | 0 | 0 | 1 | 1 | 0 | 0 | **Channel** related diagnostic, number 12 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | Channel 6 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | Upper limit, word by word |

**Figure 4-36 : Example**

## Changing the Slave Address

A request for changing the slave address is indicated through NEW\_SSA\_DATA. With the macro VPC3\_GET\_SSA\_BUF\_PTR(), a pointer to the buffer with the new slave address can be read. With the macro VPC3\_GET\_SSA\_LEN(), the user is informed of the length of the SSA buffer received.

#### Callback function:

|  |  |  |
| --- | --- | --- |
| **void DpAppl\_IsrNewSetSlaveAddress( MEM\_STRUC\_SSA\_BLOCK\_PTR psSsa )** | | |
| Function | VPC3+ has received new Set slave address telegram. | |
| Parameter | psSsa | Pointer to set slave address structure |
| Return value |  |  |

**Figure 4-37 : Function DpAppl\_IsrNewSetSlaveAddress ()**

#### Functions:

|  |  |  |
| --- | --- | --- |
| **uint8\_t VPC3\_GET\_SSA\_LEN()** | | |
| Function | Get the length of the received ssa data | |
| Parameter | None | |
| Return Value | Length of ssa data |  |

**Figure 4-38 : Function VPC3\_GET\_SSA\_LEN**

|  |  |  |
| --- | --- | --- |
| **VPC3\_UNSIGNED8\_PTR VPC3\_GET\_SSA\_BUF\_PTR ()** | | |
| Function | Fetch buffer pointer of the ssa buffer. | |
| Parameter | None | |
| Return Value | pointer to the ssa data buffer |  |

**Figure 4-39 : Function VPC3\_GET\_SSA\_BUF\_PTR**

Structure of the Set\_Slave\_Address telegram:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 0 |  |  |  |  |  |  |  |  | New\_Slave\_Address |
| 1 |  |  |  |  |  |  |  |  | Ident\_Number\_High |
| 2 |  |  |  |  |  |  |  |  | Ident\_Number\_Low |
| 3 |  |  |  |  |  |  |  |  | No\_Add\_Chg |
| 4  : 243 |  |  |  |  |  |  |  |  | Rem\_Save\_Data additional application specific data |

**Figure 4-40 : Structure of the Set\_Slave\_Address telegram**

## Global Control Commands

The interrupt New\_GC\_Command indicates the arrival of a Global\_Control message. The command VPC3\_GET\_IND\_NEW\_GC\_COMMAND supplies the Control\_Command byte. This makes it possible for the user to react to these commands. The VPC3+ internally processes these commands regarding buffer management. That is, in the case of ‘Clear’, the output data is deleted and the cleared buffer is made available to the user.

#### Callback function:

|  |  |  |
| --- | --- | --- |
| **void DpAppl\_IsrNewGlobalControlCommand( uint8\_t bGcCommand )** | | |
| Function | VPC3+ has received new global control command. | |
| Parameter | bGcCommand | Global control command |
| Return value |  |  |

**Figure 4-41 : Function DpAppl\_IsrNewGlobalControlCommand ()**

#### Functions:

|  |  |  |
| --- | --- | --- |
| **VPC3\_UNSIGNED8\_PTR VPC3\_GET\_GC\_COMMAND ()** | | |
| Function | Fetch global control byte. | |
| Parameter | None | |
| Return Value | Global control byte |  |

**Figure 4-42 : Function VPC3\_GET\_GC\_COMMAND**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Address** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 3CH | Reserved | Reserved | Sync | Unsync | Freeze | Unfreeze | Clear\_Data | Reserved | R\_GC\_ Command  See coding below |

|  |  |
| --- | --- |
|  | **R\_GC\_Command, Address 3CH:** |
| Bit 7-6 | **Reserved** |
| Bit 5 | **Sync:**  The output data transferred with a WRITE\_READ\_DATA telegram is changed from ‘D’ to ‘N.’ The following transferred output data is kept in ‘D’ until the next ‘Sync’ command is issued. |
| Bit 4 | **Unsync:**  The ‘Unsync’ command cancels the ‘Sync’ command. |
| Bit 3 | **Freeze:**  The input data is fetched from ‘N’ to ‘D’ and ‘frozen’. New input data is not fetched again until the master sends the next ‘Freeze’ command. |
| Bit 2 | **Unfreeze:**  The ‘Unfreeze’ command cancels the ‘Freeze’ command. |
| Bit 1 | **Clear\_Data:**  With this command, the output data is deleted in ‘D’ and is changed to ‘N’. |
| Bit 0 | **Reserved** |

**Figure 4-43 : Description GC\_COMMAND**

## Watchdog Timeout in DP-Control

The interrupt VPC3\_INT\_DP\_WD\_TIMEOUT indicates that the slave lost bus communication to the master. The following command returns the status of the watchdog state machine.

|  |  |  |
| --- | --- | --- |
| **uint8\_t VPC3\_GET\_WD\_STATE()** | | |
| Function | Get the Wactdog State. | |
| Parameter | None | |
| Return Value | Watchdog State |  |

**Figure 4-44 : Function VPC3\_GET\_WD\_STATE()**

|  |  |
| --- | --- |
| **Watchdog State** | **Description** |
| BAUD\_SEARCH | Baudrate search |
| BAUD\_CONTROL | Monitoring the baudrate |
| DP\_MODE | DP\_Mode; that is, bus watchdog activated |

**Figure 4-45 : Description Wachdog State**

### Leaving the Data Exchange State

The VPC3\_INT\_GO\_LEAVE\_DATA\_EX message indicates that the VPC3+ made a state change in the internal state machine.

With the following command the application is informed whether the VPC3+ has entered the data exchange state or left it. The cause for this transition can be a faulty parameter assignment message in the data transfer phase, for example.

#### Callback function:

|  |  |  |
| --- | --- | --- |
| **void DpAppl\_IsrGoLeaveDataExchange( uint8\_t bDpState )** | | |
| Function | VPC3+ has received new profibus state. | |
| Parameter | bDpState | State of profibus connection |
| Return value |  |  |

**Figure 4-46 : Function DpAppl\_IsrGoLeaveDataExchange()**

#### Functions:

|  |  |  |
| --- | --- | --- |
| **uint8\_t VPC3\_GET\_DP\_STATE()** | | |
| Function | Get the DP State. | |
| Parameter | None | |
| Return Value | DP State |  |

**Figure 4-47 : Function VPC3\_GET\_DP\_STATE()**

States of the DP-State Machine:

|  |  |
| --- | --- |
| **DP- State** | **Description** |
| WAIT\_PRM | Wait for parameter assignment |
| WAIT\_CFG | Wait for configuration |
| DATA\_EX | Data exchange |
| DP\_ERROR | Error |

**Figure 4-48 : DP States**

## VPC3\_Reset (Go\_Offline)

With the command **VPC3\_GO\_OFFLINE()** the VPC3+ enters the offline state, after the actual request is processed. The command **VPC3\_GET\_OFF\_PASS()** determines whether the transition to offline was made. If the return value is ‘zero’, the VPC3+ is ‘Offline’. If the return value is 1, the VPC3+ is ‘Passiv Idle’.

## Leave Master

The command **VPC3\_SET\_USER\_LEAVE\_MASTER()** causes the VPC3+ to change into the state ‘Wait\_Prm’.

## FatalError (DP+MSAC\_C1+MSAC\_C2)

The firmware calls this function if a grave error occurs that does not permit continuing useful processing. If the firmware calls this function, this indicates a software error in the user program. This function is not to return to the firmware!

|  |  |  |  |
| --- | --- | --- | --- |
| **FatalError** |  |  | **Grave Error** |
| Transfer | File Line  Errcb\_ptr | DP\_ERROR\_FILE  uint16\_t VPC3\_ERRCB\_PTR | Filename Source code line Specific Error |
| Return |  |  | **Function must not return!** |

**Figure 4-49 : Function Fatal\_ERROR**

|  |  |  |
| --- | --- | --- |
| **DP\_ERROR\_FILE** |  |  |
| DP\_USER | 0x10 |  |
| DP\_IF | 0x20 |  |
| DP\_ISR | 0x30 |  |
| DP\_FDL | 0x40 |  |
| DP\_C1 | 0x50 |  |
| DP\_C2 | 0x60 |  |

**Figure 4-50 : Description DP\_ERROR\_FILE**

## Notes:

## Functional Description of the DPV1 Services

When the firmware is initialized, the DPV1 services are initialized also. If the DPV1 indications are to be processed in the polling mode, the application program has to cyclically call the macros VPC3\_POLL\_IND\_FDL\_IND() and VPC3\_POLL\_IND\_POLL\_END\_IND() in

the main loop. If the DPV1 indications are to be processed in the interrupt mode, the application program has to call the macros VPC3\_GET\_IND\_FDL\_IND() and VPC3\_GET\_IND\_POLL\_END\_IND() in

the interrupt routine.

### Initiate (MSAC\_C2)

 In the answer to an Initiate REQ PDU (on SAP 49), the firmware sends a free SAP (0..48) in the immediate response. This SAP (**S**ervice **A**ccess **P**oint) has been made available previously as response.

 The RM (**R**esource **M**anager) searches for a new free SAP, and makes it available as next response for SAP 49.

 The firmware calls the function msac\_c2\_initiate\_req. The SAP that is to be used is transferred as parameter. In the function msac\_c2\_initiate\_req, the application program can check the API and SCL, for example.

 If msac\_c2\_initiate\_req was acknowledged positive, the SAP is marked as assigned.

 The SAP used is opened; via this SAP, the Initiate RES PDU is transmitted.

### Abort (MSAC\_C2)

The cancellation can be activated either by the local user via a function or via the response data, or by the master via a message.

The FW closes the communication SAP The SAP is marked as free



The function msac\_c2\_abort\_ind is called. This only happens if the user has not requested a cancellation.

### Read (MSAC\_C1 and MSAC\_C2)

 The firmware package calls the function dpv1\_read\_req as soon as a Read.req was received.

 If the data has been made available, or if an error was signalled, the reply is sent to the master.

### Write (MASC\_C1 and MSAC\_C2)

 The firmware package calls the function dpv1\_write\_req as soon as a Write.req was received

 If the data has been processed, or if an error was signaled, the reply is sent to the master.

### Data Transport (MSAC\_C2)

 The firmware package calls the function msac\_c2\_data\_transport\_req as soon as a Data\_Transport.req was received.

 If the response data was made available, or if an error was signaled, the reply is sent to the master.

### Diagnosis, Alarms, and Status Messages in the case of DPV1

In DPV1, an alarm- and status model is defined. The alarms and status messages are transmitted via a device-related diagnosis. For that reason, The DPV1 slave is to use the device-related diagnoses only in this sense. The alarm is acknowledged by the master and the user enter the alarm diagnostic to the alarm state machine. The status message isn’t acknowledge by the master. The user set the status message directly in the diagnostic buffer. The DPV1 slave can continue using the id-related and channel-related diagnoses, as described in the DP standard. The application program may write to the diagnostic data as is the case with the DP slave. In addition, the user can enter status messages in the diagnostic buffer. In DPV1, the static diagnosis has a special meaning: with static diagnosis, the slave signals that it is logically not ready to make useful data available. This is the case, for example, if a sensor was correctly parameterized and configured, but has not yet been set to its measuring range via the MSAC\_C1 channel. If the slave can supply useful data, it removes the static diagnosis.

### Error Handling

If the application detects an error while processing a user function, it writes the Error Code 1 and 2 according to the structure below to the response buffer that was transferred to it previously, and returns the value DPV1\_NOK. The firmware fills in the function number and the decode field.

|  |  |  |
| --- | --- | --- |
| **DPV1\_NEG\_RES\_PDU** |  | **Error Response Block** |
| Function\_num | uint8\_t | Is entered by the firmware |
| Err\_decode | uint8\_t | Always DPV1\_ERR\_DEC\_DPE |
| Err\_code1 | uint8\_t | DPV1 Error Code |
| Err\_code2 | uint8\_t | User-specific |

**Figure 5-1 : Error Response Block**

Bit-No

0

1

2

3

4

5

6

7

**Error Code Error Class**

**Figure 5-2 : Error Code / Error Class**

|  |  |  |
| --- | --- | --- |
| **Error\_Class** | **Meaning** | **Error\_Code** |
| 0 to 9 | Reserved \*) |  |
| 10 | Application | 0 = read error  1 = write error  2 = module failure  3 to 7 = reserved \*)  8 = version conflict  9 = feature not supported  10 to 15 = user specific |
| 11 | Access | 0 = invalid index  1 = write length error  2 = invalid slot  3 = type conflict  4 = invalid area  5 = state conflict  6 = access denied  7 = invalid range  8 = invalid parameter  9 = invalid type  10 to 15 = user specific |
| 12 | Resource | 0 = read constrain conflict  1 = write constrain conflict  2 = resource busy  3 = resource unavailable  4 to 7 = reserved \*)  8 to 15 = user specific |
| 13 to 15 | User specific |  |

**Figure 5-3 : Error Code / Error Class**

\*) Reserved Error\_Codes are intended to be passed unchanged to the user.

Defines for Error Code / Error Class in the firmware:

|  |  |  |
| --- | --- | --- |
| **Error Class** |  |  |
| Reserved | 0 – 9 | Reserved |
| DPV1\_ERRCL\_APPLICATION | 10 | Error on application level |
| DPV1\_ERRCL\_ACCESS | 11 | Access error |
| DPV1\_ERRCL\_RESSOURCE | 12 | Resource error |
| DPV1\_ERRCL\_USER | 13 (-15) | Free for application |

**Figure 5-4 : Error Class**

|  |  |  |
| --- | --- | --- |
| **Error\_Code for Error\_Class DPV1\_ERRCL\_APPLICATION** |  |  |
| DPV1\_ERRCL\_APP\_READ | 0 | Read error |
| DPV1\_ERRCL\_APP\_WRITE | 1 | Write error |
| DPV1\_ERRCL\_APP\_MODULE | 2 | Module error |
| Reserved | 3-7 | reserved |
| DPV1\_ERRCL\_APP\_VERSION | 8 | Version conflict |
| DPV1\_ERRCL\_APP\_NOTSUPP | 9 | Not supported |
| DPV1\_ERRCL\_APP\_USER | 10 (-15) | Free for application |

**Figure 5-5 : Error Code for Application Error Class**

|  |  |  |
| --- | --- | --- |
| **Error\_Code for Error\_Class DPV1\_ERRCL\_ACCESS** |  |  |
| DPV1\_ERRCL\_ACC\_INV\_INDEX | 0 | Impermissible index |
| DPV1\_ERRCL\_ACC\_WRITE\_LEN | 1 | Write length wrong |
| DPV1\_ERRCL\_ACC\_INV\_SLOT | 2 | Impermissible slot |
| DPV1\_ERRCL\_ACC\_TYPE | 3 | Type conflict |
| DPV1\_ERRCL\_ACC\_INV\_AEREA | 4 | Impermissible area |
| DPV1\_ERRCL\_ACC\_STATE | 5 | State conflict |
| DPV1\_ERRCL\_ACC\_ACCESS | 6 | Access not permitted |
| DPV1\_ERRCL\_ACC\_INV\_RANGE | 7 | Impermissible range |
| DPV1\_ERRCL\_ACC\_INV\_PARAM | 8 | Impermissible parameter |
| DPV1\_ERRCL\_ACC\_INV\_TYPE | 9 | Impermissible type |
| DPV1\_ERRCL\_ACC\_USER | 10 (-15) | Free for application |

**Figure 5-6 : Error Code for Access Error Class**

|  |  |  |
| --- | --- | --- |
| **Error\_Code for Error\_Class DPV1\_ERRCL\_RESOURCE** |  |  |
| DPV1\_ERRCL\_RES\_READ\_CONSTRAIN | 0 | Read constrain conflict |
| DPV1\_ERRCL\_RES\_WRITE\_CONSTRAIN | 1 | Write constrain conflict |
| DPV1\_ERRCL\_RES\_BUSY | 2 | Resource busy |
| DPV1\_ERRCL\_RES\_UNAVAIL | 3 | Resource unavailable |
| Reserved | 4 – 7 | reserved |
| DPV1\_ERRCL\_RES\_USER | 8 (- 15) | Free for application |

**Figure 5-7 : Error Code for Resource Error Class**

## Initialization

### Settings for DPV1 in the DpCfg.h

The user connects the different services via #define in “cfg.h”, so that the program code is adapted to the required services respectively.

|  |  |
| --- | --- |
| **Service** |  |
| #define DP\_MSAC\_C1 | 1: Activation of the functionality for the expansion services of the Class 1 master.  0: not activated |
| #define DP\_MSAC\_C2 | 1: Activation of the functionality for the expansion services of the Class 2 master.  0: not activated |
| #define DP\_ALARM | 1: Activation of the functionality for the expansion services of the alarm mode.  0: not activated |
| #define DPV1\_IM\_SUPP | 1: Activation of the functionality for the expansion services of the I&M functionality.  0: not activated |

**Figure 5-8 : PROFIBUS Services**

|  |  |  |
| --- | --- | --- |
| **Settings for MSAC\_C2 Service** |  |  |
| #define DP\_MSAC\_C2\_Time |  | Enables timecontrol for C2 services |
| #define C2\_NUM\_SAPS | uint8\_t | Number of SAPs that the firmware makes available for MSAC\_C2 Connections |
| #define C2\_LEN | uint8\_t | MSAC\_C2 PDU length of the C2-SAP (20...244) |
| #define C2\_FEATURES\_SUPPORTED\_1 | uint8\_t | = 0x01 (MSAC\_C2\_READ and  MSAC\_C2\_WRITE supported) |
| #define C2\_FEATURES\_SUPPORTED\_2 | uint8\_t | = 0x00 |
| #define C2\_PROFILE\_FEATURES\_1 | uint8\_t | Profile or vendor specific |
| #define C2\_PROFILE\_FEATURES\_2 | uint8\_t | Profile or vendor specific |
| #define C2\_PROFILE\_NUMBER | uint16\_t | Profile or vendor specific |

**Figure 5-9 : Settings for MSAC\_C2 Service**

|  |  |  |
| --- | --- | --- |
| **Settings for MSAC\_C1 Service** |  |  |
| #define C1\_LEN | uint8\_t | Length of MSAC\_C1 Data (4..244 Bytes) |

**Figure 5-10 : Settings for MSAC\_C1**

|  |  |
| --- | --- |
| **Settings for MSAC\_C1 Alarm** |  |
| #define DP\_ALARM\_OVER\_SAP50 | 1: The master handles the Alarm Acknowledge over SAP 50  0: The master handles the Alarm Acknowledge over SAP 51 |

**Figure 5-11 : Settings for MSAC\_C1\_Alarm**

#### Mandatory settings in the VPC3+:

|  |  |
| --- | --- |
| **Mode Register 0, High-Byte, Address 07H (Intel):** | |
| Bit 10 | **User\_Time\_Base:** Timebase of the cyclical User\_Time\_Clock-Interrupt  0 = The User\_Time\_Clock-Interrupt occurs every 1 ms.  **1 = The User\_Time\_Clock-Interrupt occurs every 10 ms. (mandatory DPV1)** |

**Figure 5-12 : Mode Register**

#### Enable following interrupts:

|  |  |
| --- | --- |
| **Interrupt-Mask-Register, Low-Byte, Address 04H (Intel):** | |
| Bit 4 | **User\_Timer\_Clock:**  The time base for the User\_Timer\_Clocks has run out ( 1 /10ms). |
| Bit 2 | **Baudrate\_Detect:**  The VPC3+ has left the ‘Baud\_Search state’ and found a baud rate. |

**Figure 5-13 : Interrupt Mask Register**

|  |  |
| --- | --- |
| **Interrupt Mask Register 0, High-Byte, Address 05H (Intel):** | |
| Bit 15 | **FDL\_Ind:**  The VPC 3+ has received an acyclic service request and made the data available in an indication buffer. |
| Bit 14 | **Poll\_End\_Ind:**  The VPC 3+ have send the response to an acyclic service. |

**Figure 5-14 : Interrupt Mask Register**

During the initialization the SAP-list will be generated (dp\_fdl.c). Each entry in the SAP list consist of 7 bytes. The pointer at address 17H contains the segment base address of the first element of the SAP list. The last element in the list is always indicated with FFH. If the SAP list shall not be used, the first entry must be FFH, so the pointer at address 17H must point to a segment base address location which contains FFH.

#### The MSAC\_C2 service is enabled after VPC3\_START() and the MSAC\_C1 is enabled with DPV1\_Enable in the Set\_Param telegram.

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Master SAP** | **Slave SAP** | **Service** |
| MSAC\_C1 | 51 | 50 or 51 | Alarm\_Ack |
| MSAC\_C1 | 51 | 51 | READ/WRITE |
| MSAC\_C2 | 50 | 49 | Initiate.req |
| MSAC\_C2 | 50 | 48 .. 0 | Abort, Read/Write, Data\_Transfer |

**Figure 5-15 : SAPs for acyclic services**

#### Structure of SAP-List entry:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 0 |  |  |  |  |  |  |  |  | SAP\_Number |
| 1 |  |  |  |  |  |  |  |  | Request\_SA |
| 2 |  |  |  |  |  |  |  |  | Request\_SSAP |
| 3 |  |  |  |  |  |  |  |  | Service\_Supported |
| 4 |  |  |  |  |  |  |  |  | Ind\_Buf\_Ptr[0] |
| 5 |  |  |  |  |  |  |  |  | Ind\_Buf\_Ptr[1] |
| 6 |  |  |  |  |  |  |  |  | Resp\_Buf\_Ptr |

|  |  |
| --- | --- |
| **SAP-List entry:** | |
| Byte 0 | **Response\_Sent:** Response-Buffer sent  0 = no Response sent  1 = Response sent  **SAP\_Number:** 0 – 63  In DP-Mode the SAPs 53, 55-62 are used for cyclic communication. |
| Byte 1 | **Request\_SA:** The source address of a request is compared with this value. At differences, the VPC 3+ response with No-Service-Activated (RS). The default value for this entry is 7FH. |
| Byte 2 | **Request\_SSAP:** The source SAP of a request is compared with this value. At differences, the VPC 3+ response with No-Service-Activated (RS). The default value for this entry is 7FH. |
| Byte 3 | **Service\_Supported:** Indicates the permitted FDL service. 00 = all FDL services allowed |
| Byte 4 | **Ind\_Buf\_Ptr[0]:** pointer to indication buffer 0 |
| Byte 5 | **Ind\_Buf\_Ptr[1]:** pointer to indication buffer 1 |
| Byte 6 | **Resp\_Buf\_Ptr:** pointer to response buffer |

**Figure 5-16 : SAP list entry**

#### Example of SAP-list:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SAP** |  |  |  |  |  |  | **Service** |
| 31 | 7F | 7F | 0B | 5C | 5C | 5B | Initiate\_Req (Resource Manager) |
| 30 | 07 | 7F | 0B | 5C | 5C | 5C | MSAC\_C2 channel 1 |
| 2F | 07 | 7F | 0B | 63 | 63 | 63 | MSAC\_C2 channel 2 |
| 33 | 7F | 7F | 0B | 6A | 6A | 6A | MSAC\_C1 channel |
| FF | 00 | 00 | 00 |  |  |  |  |

**Figure 5-17 : Example of SAP list (after START\_VPC3())**

In addition an indication and response buffers are needed. Each buffer consists of a 4 byte header for the buffer management and a data block of configurable length.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 0 | USER | IND | RESP | INUSE |  |  |  |  | Control |
| 1 |  |  |  |  |  |  |  |  | Max\_Length |
| 2 |  |  |  |  |  |  |  |  | Length |
| 3 |  |  |  |  |  |  |  |  | Function Code |

|  |  |
| --- | --- |
|  | **SAP-List entry:** |
| Byte 0 | **Control:** bits for buffer management USER buffer assigned to user  IND indication data included in buffer RESP response data included in buffer INUSE buffer assigned to VPC 3+ |
| Byte 1 | **Max\_Length:** length of buffer |
| Byte 2 | **Length:** length of data included in buffer |
| Byte 3 | **Function Code:** function code of the telegram |

**Figure 5-18 : Buffer Header**

## DP-V1 Callback Functions

Callback functions are functions that the DPV1 state machine has to make available for the user application. Via the return value, the user controls whether he has completed the function successful, or whether he has completed the function with error, or he wanted to cancel the connection. The callback functions are handled in the file DpV1.c.

|  |  |
| --- | --- |
| **Return Values of the Callback Functions** | |
| DPV1\_OK | The function was completed successfully |
| DPV1\_NOK | An error occurred. The user entered more detailed information about the error in the error block for this channel (refer to chapter Error Handling). |
| DPV1\_DELAY | The application program is processing a request asynchronously. |
| DPV1\_ABORT | The user wants to cancel the affected C2 connection. Previously, the user has preprocessed the abort PDU in the ASIC memory area. |

**Figure 5-19 : Return Value of Callback Function**

Which return values are permitted respectively is provided with the individual functions.

### Dpv1\_Msac2InitiateReq (MSAC\_C2)

The firmware calls this functon if a master wants to establish a MSAC\_C2 connection.

|  |  |  |
| --- | --- | --- |
| **DPV1\_RET\_VAL Dpv1\_Msac2InitiateReq( uint8\_t bSapNr, INITIATE\_REQ\_PDU\_PTR, MSG\_HEADER\_PTR psMsgHeader,VPC3\_UNSIGNED8\_PTR pToDpv1Data )** | | |
| Function | Establish a C2 connection | |
| Parameter | bSapNr | Address of the slave |
| psInitiateReq | Local copy of Initiate.req telegram |
| psMsgHeader | Pointer to message header |
| pToDpv1Data | Pointer to DPV1 data |
| Return Value | DPV1\_OK DPV1\_NOK DPV1\_DLAY | See DPV1\_RET\_VAL |

**Figure 5-20 : Function Dpv1\_Msac2\_InitiateReq**

When this function is called, the parameter PDU points to the structure MSAC\_C2\_INITIATE\_REQ\_PDU. When leaving the function, the user program has to have preprocessed the buffer according to the structure MSAC\_C2\_INITIATE\_RES\_PDU. The user is supported with the function MSAC\_C2\_INITIATE\_REQ\_TO\_RES; it generates the response structure from the request structure. This applies only if the slave is the endpoint of the connection. If the macro MSAC\_C2\_INITIATE\_REQ\_TO\_RES returns the value DPV1\_NOK, the PDU that was received remains unchanged. The user has to either make the evaluation himself, or reject the request for establishing a connection.

The firmware sends the response PDU when the application program leaves the function with DPV1\_OK. If the application program can’t establish the connection (for example, profile is not supported), the application program has to fill in the response PDU according to the structure DPV1\_ABORT\_PDU, and exit the function with DPV1\_ABORT. The firmware will then set the correct function number, and send the PDU as response. In this case, the firmware does not open the connection, and marks the corresponding SAP as free again. The request for establishing a connection may also be refused with negative response data (DPV1\_ERROR\_RES).

#### Comment: The application is not to change the function number received.

|  |  |  |
| --- | --- | --- |
| **DPV1\_INITIATE\_REQ\_PDU** |  | **Initiate Request Structure** |
| function\_num | uint8\_t | 0x57 |
| reserved1 | uint8\_t | Reserved byte |
| reserved2 | uint8\_t | Reserved byte |
| reserved3 | uint8\_t | Reserved byte |
| send\_timeout | uint16\_t | Time control for MSAC\_C2 |
| features\_supported1 | uint8\_t | 0x01 (Read/Write service) |
| features\_supported2 | uint8\_t | Reserved |
| profile\_features\_supported1 | uint8\_t | Profile-,vendor specific |
| profile\_features\_supported2 | uint8\_t | Profile-,vendor specific |
| profile\_ident\_number | uint16\_t | Vendor specific |
| s\_type | uint8\_t |  |
| s\_len | uint8\_t |  |
| d\_type | uint8\_t |  |
| d\_len | uint8\_t |  |
| addr\_data | BYTE[s\_len +  d\_len] | Structure according to  DPV1\_INITIATE\_SUB\_PARAM |

**Figure 5-21 : Structure DPV1\_INITIATE\_REQUEST**

#### S-Type:

This subparameter indicates the presence (S-Type=1) of the optional Network/MAC address in the Add\_Addr\_Param of the source.

#### S-Len:

This subparameter indicates the length of the S\_Addr subparameter.

#### D-Type:

This subparameter indicates the presence (D-Type=1) of the optional Network/MAC address in the Add\_Addr\_Param of the destination.

#### D-Len:

This subparameter indicates the length of the D\_Addr subparameter.

#### addr\_data:

Contains the additional address information of the source and of the destination.

|  |  |  |
| --- | --- | --- |
| **DPV1\_INITIATE\_RES\_PDU** |  | **Initiate Response Structure** |
| function\_num | uint8\_t | 0x57 |
| max\_len\_data\_unit | uint8\_t | Length data unit |
| features\_supported1 | uint8\_t | 0x01 (Read/Write service) |
| features\_supported2 | uint8\_t | Reserved |

|  |  |  |
| --- | --- | --- |
| profile\_features\_supported1 | uint8\_t | Profile-,vendor specific |
| profile\_features\_supported2 | uint8\_t | Profile-,vendor specific |
| profile\_ident\_number | uint16\_t | Vendor specific |
| s\_type | uint8\_t | See above |
| s\_len | uint8\_t | See above |
| d\_type | uint8\_t | See above |
| d\_len | uint8\_t | See above |
| addr\_data | BYTE[s\_len +  d\_len] | Structure according to  DPV1\_INITIATE\_SUB\_PARAM |

**Figure 5-22 : Structure DPV1\_INITIATE\_RESPONSE**

|  |  |  |
| --- | --- | --- |
| **addr\_data[]** |  |  |
| S\_api | uint8\_t |  |
| S\_reserved | uint8\_t |  |
| S\_net\_addr | uint8\_t[6] |  |
| S\_mac\_addr | uint8\_t[] |  |
| D\_api | uint8\_t |  |
| D\_reserved | uint8\_t |  |
| D\_net\_addr | uint8\_t[6] |  |
| D\_mac\_addr | uint8\_t[] |  |

**Figure 5-23 : Structure addr\_data**

#### S\_API:

This subparameter identifies the application process instance of the source.

#### S\_Network\_Address: (S-Type=1)

This subparameter identifies the network address of the source according to ISO/OSI-Network addresses.

#### S\_MAC\_Address: (S-Type=1)

This subparameter identifies the MAC\_Address of the source.

#### D\_api:

This subparameter identifies the application process instance of the destination.

#### D\_Network\_Address: (D-Type=1)

This subparameter identifies the network address of the destination according to ISO/OSI-Network addresses.

#### D\_MAC\_Address: (D-Type=1)

This subparameter identifies the MAC\_Address of the destination.

### MSAC\_C2\_INITIATE\_REQ\_TO\_RES (MSAC\_C2)

This function relieves the application program of copying the data that is located at different locations at the initiate request and the response PDU. In addition, standard settings are entered in the response PDU.

|  |  |  |  |
| --- | --- | --- | --- |
| **MSAC\_C2\_INITIATE\_REQ\_TO\_RES** | | | |
| Transfer | PDU | MSAC\_C2\_INITIATE\_REQ\_PD  U \* | Request PDU |
| Return | DPV1\_OK DPV1\_NOK |  | Response PDU was generated  The user has to handle the Response PDU himself since the device is not the endpoint of the connection. The PDU that has been transferred is not changed. |

**Figure 5-24 : Function MSAC\_C2\_INITIATE\_REQ\_TO\_RES**

#### Function Description:

 A check is made in the connection buffer whether the endpoint (D type = 0) of a connection has been reached. Only then will the response PDU be generated; that is, the buffer that was received is changed.

 The following response PDU is generated:

* + - * As length for the PDU, the length entry for the MSAC\_C2 PDU transferred with vpc3\_init() is used.
      * Only READ and WRITE is specified for supported services
      * The profile attributes and the profile number are set to default values (defined in dp\_cfg.h).
      * The data for destination- and source addressing is copied from the request PDU and entered in the response PDU; destination and source are exchanged.

### Dpv1\_Msac2AbortInd

The firmware calls this function if a MSAC\_C2 connection was aborted by the master, or the firmware detects a reason for canceling it (for example, timeout). A MSAC\_C1 connection is coupled to the processing mode (cyclical state machine) of the slave. In the case of LEAVE\_DATA\_EXCHANGE, the MSAC\_C1 connection is cancelled automatically.

|  |  |  |  |
| --- | --- | --- | --- |
| **USER\_C2\_ ABORT\_IND** | | **ABORT Indication Callback Function** | |
| Transfer | SAP PDU | uint8\_t DPV1\_PTR \* | SAP number |
| Return | DPV1\_OK |  | See above |

**Figure 5-25 : Function USER\_C2\_ABORT\_IND**

|  |  |  |
| --- | --- | --- |
| **DPV1\_ABORT\_PDU** |  | **Abort Structure** |
| function\_num | uint8\_t |  |
| Subnet | uint8\_t |  |
| instance\_reason | uint8\_t |  |

**Figure 5-26 : Function DPV1\_ABORT\_PDU**

|  |  |  |
| --- | --- | --- |
| **Subnet** |  |  |
| MSAC\_C2\_SUBNET\_NO | 0 |  |
| MSAC\_C2\_SUBNET\_LOCAL | 1 |  |
| MSAC\_C2\_SUBNET\_REMOTE | 2 |  |

**Figure 5-27 : Description Subnet**

|  |  |  |
| --- | --- | --- |
| **Instance** |  |  |
| MSAC\_C2\_INSTANCE\_FDL | 0x00 |  |
| MSAC\_C2\_INSTANCE\_MSAC\_C2 | 0x10 |  |
| MSAC\_C2\_INSTANCE\_USER | 0x20 |  |
| MSAC\_C2\_INSTANCE\_RESERVED | 0x30 |  |

**Figure 5-28 : Description Instance**

|  |  |  |
| --- | --- | --- |
| **reason** |  |  |
| MSAC\_C2\_ABT\_SE | 0x01 | Sequence error |
| MSAC\_C2\_ABT\_FE | 0x02 | Invalid request PDU received |
| MSAC\_C2\_ABT\_TO | 0x03 | Timeout of the connection |
| MSAC\_C2\_ABT\_RE | 0x04 | Invalid response PDU received |
| MSAC\_C2\_ABT\_IV | 0x05 | Invalid service from USER |
| MSAC\_C2\_ABT\_STO | 0x06 | Send\_Timeout requested was too small |
| MSAC\_C2\_ABT\_IA | 0x07 | Invalid additional address information |
| MSAC\_C2\_ABT\_OC | 0x08 | waiting for FDL\_DATA\_REPLY.con |
| MSAC\_C2\_ABT\_RES | 0x0F | Resource error |

**Figure 5-29 : Description Reason**

### Dpv1\_ReadReq (MSAC\_C1+MSAC\_C2)

The firmware calls this function when a read request is pending.

|  |  |  |
| --- | --- | --- |
| **DPV1\_RET\_VAL Dpv1\_ReadReq( uint8\_t bSapNr, MSG\_HEADER\_PTR psMsgHeader,VPC3\_UNSIGNED8\_PTR pToDpv1Data )** | | |
| Function | DP-V1 read request | |
| Parameter | bSapNr | PROFIBUS service access point |
| psMsgHeader | Pointer to message header |
| pToDpv1Data | Pointer to DPV1 data |
| Return Value | DPV1\_OK DPV1\_NOK DPV1\_DELAY | See DPV1\_RET\_VAL |

**Figure 5-30 : Function Dpv1\_ReadReq**

The firmware calls this function when a Read request has been received. The array pToDpv1Data[] is undefined when the function is called. The application program has to fill in the array pToDpv1Data[], and enter the corresponding length in the field ‘length’. The firmware handles the function number. If there is an error, the user normally provides a negative response PDU. This retains the connection. If the connection is to be cancelled also, an ABORT PDU is to be generated.

|  |  |  |
| --- | --- | --- |
| **DPV1\_READ\_PDU** |  | **Read Structure** |
| Function\_num | uint8\_t | 0x5E |
| Slot\_num | uint8\_t |  |
| Index | uint8\_t |  |
| Length | uint8\_t |  |
| Pdu\_data | uint8\_t[] |  |

**Figure 5-31 : Description DPV1\_READ\_PDU**

#### Example for Read Processing:

 Read.req(length 40) for a data set with the length 40 octets => the length indicated in the request is read

 Read.req(length > 40) for a data set with the length 40 octets => the genuine length of the data set (40 bytes) is read

### DpV1\_WriteReq (MSAC\_C1+MSAC\_C2)

The firmware calls this function if a write request was received. The firmware manages the function number. If there is an error, the user normally sets up a negative response PDU. This retains the connection. If the connection is to be cancelled also, an ABORT PDU is to be generated.

|  |  |  |
| --- | --- | --- |
| **DPV1\_RET\_VAL Dpv1\_WriteReq( uint8\_t bSapNr, MSG\_HEADER\_PTR psMsgHeader,VPC3\_UNSIGNED8\_PTR pToDpv1Data )** | | |
| Function | DP-V1 write request | |
| Parameter | bSapNr | PROFIBUS service access point |
| psMsgHeader | Pointer to message header |
| pToDpv1Data | Pointer to DPV1 data |
| Return Value | DPV1\_OK DPV1\_NOK DPV1\_DELAY | See DPV1\_RET\_VAL |

**Figure 5-32 : Function Dpv1\_WriteReq**

#### Example for Write Processing:

 Write.req(length 40) for a data set with the length 40 octets => the length of data indicated in the request is written, and the length is mirrored in the reply.

 Write.req(length > 40) for a data set with the length 40 octets => there is to be no writing; an error message has to be transmitted.

|  |  |  |
| --- | --- | --- |
| **DPV1\_WRITE\_PDU** |  | **Write Structure** |
| Function\_num | uint8\_t | 0x5F |
| Slot\_num | uint8\_t |  |
| Index | uint8\_t |  |
| Length | uint8\_t |  |
| Pdu\_data | uint8\_t[] |  |

**Figure 5-33 : Description DPV1\_WRITE\_PDU**

### Dpv1\_Msac2DataTransportReq (MSAC\_C2)

The firmware calls this function if a data transport request was received. When the function is called, the array pToDpv1Data[] contains the received data. The application program has to fill the array pToDpv1Data[] with the data that is to be sent, and set the field ‘length’ correspondingly. The firmware handles the function number. If there is an error, the user normally sets up a negative response PDU. This retains the connection. If the connection is to be cancelled also, an ABORT PDU is generated.

|  |  |  |
| --- | --- | --- |
| **DPV1\_RET\_VAL Dpv1\_Msac2DataTransportReq( uint8\_t bSapNr, MSG\_HEADER\_PTR psMsgHeader,VPC3\_UNSIGNED8\_PTR pToDpv1Data )** | | |
| Function | DP-V1 data transport request | |
| Parameter | bSapNr | PROFIBUS service access point |
| psMsgHeader | Pointer to message header |
| pToDpv1Data | Pointer to DPV1 data |
| Return Value | DPV1\_OK DPV1\_NOK DPV1\_DLAY | See DPV1\_RET\_VAL |

**Figure 5-34 : Function Dpv1\_Msac2DataTransportReq**

|  |  |  |
| --- | --- | --- |
| **DATA\_TRANSPORT\_PDU** |  | **Data Transport Structure** |
| Function\_num | uint8\_t | 0x51 |
| Slot\_num | uint8\_t |  |
| Index | uint8\_t |  |
| Length | uint8\_t |  |
| Pdu\_data | uint8\_t[] |  |

**Figure 5-35 : Description DATA\_TRANSPORT\_PDU**

## DPV1 Alarm-Handling

The alarm and status messages will be transferred within the Ext\_Diag\_Data and replaces the device related diagnosis of EN 50170. The Ext\_Diag\_Data can consist of one, multiple or all of the following components:

Alarm-PDU (only one) Status-PDU



Identification-related diagnosis Channel-related diagnosis Revision-Number (only one)

The structure of the PDUs for alarm and status is as follows:

|  |  |
| --- | --- |
| **Byte** | **Description** |
| 0 | Headerbyte |
| 1 | Alarm\_Type / Status\_Type |
| 2 | Slot\_Number |
| 3 | Specifier |
| 4  : | Diagnostic User Data |

**Figure 5-36 : Structure of the device-related diagnosis for alarm / status**

### Coding of the Alarm PDU

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | **Bit7** | **Bit6** | **Bit5** | **Bit4** | **Bit3** | **Bit2** | **Bit1** | **Bit0** |
| 0 | **0** | **0** | Block length in byte (4 to 63) | | | | | |
| 1 | **0** | Alarm Type | | | | | | |
| 2 | Slot Number | | | | | | | |
| 3 | Seq\_Nr | | | | | ACK | SPEC | |
| 4  : 62 | Diagnostic User Data | | | | | | | |

**Figure 5-37 : Alarm-Pdu**

The Alarm\_Type describes the alarm itself. The necessary reaction of the control application in the DPV1-Master (Class 1) is manufacturer- or application-specific.

|  |  |
| --- | --- |
| **Alarm Type** |  |
| 0 | Reserved |
| 1 | Diagnostic Alarm |
| 2 | Process Alarm |
| 3 | Pull Alarm |
| 4 | Plug Alarm |
| 5 | Status Alarm |
| 6 | Update Alarm |
| 7-31 | Reserved |
| 32-126 | Manufacturer specific Alarm |
| 127 | Reserved |

**Figure 5-38 : Coding Alarm Type**

#### Alarm\_specifier:

|  |  |  |
| --- | --- | --- |
| **Coding** | **Designation** |  |
| 00 | No further differentiation |  |
| 01 | Error appears and Slot disturbed | the slot generates an alarm due to an error |
| 10 | Error disappears and Slot is okay | the slot generates an alarm and indicates that the slot has no further errors |
| 11 | Error disappears and Slot is still disturbed | the slot generates an alarm and indicates that the slot has still further errors |

**Figure 5-39 : Coding Alarm Specifier**

#### Add\_Ack:

When setting this bit the slave indicates to the DPV1-Master (Class 1) that this alarm requires in addition to the MSAC1\_Alarm\_Ack a separate user acknowledgement. This can be done for instance by means of a Write service.

#### Seq\_Nr:

By means of the Seq\_Nr an unique identification of an alarm message is accomplished.

### Coding of the Status PDU

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | **Bit7** | **Bit6** | **Bit5** | **Bit4** | **Bit3** | **Bit2** | **Bit1** | **Bit0** |
| 0 | **0** | **0** | Block length in byte (4 to 63) | | | | | |
| 1 | **1** | Status Type | | | | | | |
| 2 | Slot Number | | | | | | | |
| 3 | reserved | | | | | | SPEC | |
| 4  : 62 | Diagnostic User Data | | | | | | | |

**Figure 5-40 : Status Pdu**

|  |  |
| --- | --- |
| **Status Type** |  |
| 0 | Reserved |
| 1 | Status Message |
| 2 | Modul Status |
| 3-31 | Reserved |
| 32-126 | Manufacturer specific Status |
| 127 | Reserved |

**Figure 5-41 : Coding Status Type**

#### Status\_specifier:

|  |  |
| --- | --- |
| **Coding** | **Designation** |
| 00 | No further differentiation |
| 01 | Status appears |
| 10 | Status disappears |
| 11 | Reserved |

**Figure 5-42 : Coding Status Specifier**

### Coding of Modul Status

The Modul\_Status contains information whether the modules/slots of a DPV1-Slave delivers valid data or not and the information whether there is a wrong module or no module in place. For each module/slot 2 bits are designated. The Modul\_Status is padded to byte limits and not used bits are fixed to zero. The Modul\_Status is typically generated by the device module (Slot\_Number = 0).

#### Structure of the Modul\_Status:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | **Bit7** | **Bit6** | **Bit5** | **Bit4** | **Bit3** | **Bit2** | **Bit1** | **Bit0** |
| 0 | Headerbyte | | | | | | | |
| 1 | Status\_Type = Modul\_Status | | | | | | | |
| 2 | Slot Number = 0 | | | | | | | |
| 3 | Specifier | | | | | | | |
| 4 | Modul\_Status 4 | | Modul\_Status 3 | | Modul\_Status 2 | | Modul\_Status 1 | |
| : | ..... | | | | | | | |
| m | Modul\_Status m | | Modul\_Status m-1 | | . | | . | |

**Figure 5-43 : Structure Modul Status**

#### Modul Status:

|  |  |
| --- | --- |
| **Coding** | **Designation** |
| 00 | data valid |
| 01 | data invalid: the data of the corresponding module are not valid due to an error (e.g. short circuit) |
| 10 | data invalid/wrong module: the data of the corresponding module are not valid, due to a wrong module in place |
| 11 | data invalid/no module: the data of the corresponding module are not valid, because there is no module in place |

**Figure 5-44 : Coding Modul Status**

### Example for Ext\_Diag\_Data (Alarm and Status PDU)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MSB LSB** | | | | | | | |  |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |
| **0** | **0** | 0 | 0 | 0 | 1 | 1 | 1 | **Header: Device** related diagnostic |
| **1** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Statustype: Status Message |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | Slotnumber: 2 (sensor A) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Specifier: no further differentiation |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | Diag. User Data: average temperature |
|  |  |  |  |  |  |  |  | Temperature value |
|  |  |  |  |  |  |  |  | Unsigned16 |
| **0** | **0** | 0 | 0 | 1 | 0 | 0 | 1 | **Header: Device** related diagnostic |
| **0** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | Alarmtype: Process Alarm |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | Slotnumber: 3 (valve B) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Specifier: alarm appears |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | Diag. User Data: 0x50 (upper limit ex...) |
|  |  |  |  |  |  |  |  | Time stamp |
|  |  |  |  |  |  |  |  | 4 bytes |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| **0** | **1** | 0 | 0 | 0 | 0 | 1 | 0 | **Header: Identification** related diagn. |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1st Identification number with diagn. |

**Figure 5-45 : Example**

#### Correspondending GSD-part:

;text assignments for sensor A and valve B Unit\_Diag\_Area = 24-27

Value(1) = "Minimum temperature" Value(2) = "Maximum temperature" Value(5) = "Average temperature"

Unit\_Diag\_Area\_End

Unit\_Diag\_Area = 28-31

Value(1) = "lower limit exceeded pressure" Value(5) = "upper limit exceeded pressure"

Unit\_Diag\_Area\_End

Unit\_Diag\_Area = 8-15 Value(2) = "senor A" Value(3) = "valve B"

Unit\_Diag\_Area\_End

Unit\_Diag\_Area = 16-17

Value(1) = "alarm/status appearing" Value(2) = "alarm/status disappearing"

Unit\_Diag\_Area\_End

Since these definitions are used for both alarms and status messages their values should be different. That means different values for alarms and status messages should be used at the same position within the diagnostic field.

### Coding of the Alarm\_Ack-PDU

|  |  |  |
| --- | --- | --- |
| **ALARM\_ACK\_PDU** |  |  |
| Function\_num | uint8\_t | 0x5C |
| Slot\_num | uint8\_t |  |
| Alarmtype | uint8\_t |  |
| Specifier | uint8\_t |  |
| Seq\_Nr | uint8\_t[] |  |

**Figure 5-46 : Description ALARM\_ACK\_PDU**

### Alarm User Callback Functions

**VPC3\_SetAlarm**

|  |  |  |
| --- | --- | --- |
| **uint8\_t VPC3\_SetAlarm( ALARM\_STATUS\_PDU\_PTR psAlarm, uint8\_t bCallback)** | | |
| Function | By calling this function, the user can send alarms to the master | |
| Parameter | psAlarm | Pointer to alarm structure |
| bCallback | 0:the stack sends directly alarm data  1:the stack calls the function DpDiag\_Alarm |
| Return Value | SET\_ALARM\_OK |  |
| SET\_ALARM\_AL\_STATE  \_CLOSED | Alarm state machine not started |
| SET\_ALARM\_ALARMTY PE\_NOTSUPP | Alarm type not supported |
| SET\_ALARM\_SEQ\_NR\_ ERROR | The values of the transfer parameters are not in the specified value range |
| SET\_ALARM\_SPECIFIER  \_ERROR | The values of the transfer parameters are not in the specified value range |

**Figure 5-50 : Function VPC3\_SetAlarm ()**

If the parameter callback is “FALSE” the alarm will be send directly. If the parameter callback is “TRUE” the alarm will be send over the function user\_alarm (dp\_user.c). In this function the user can add e.g. ModuleStatus or Channel related diagnostic.

### Acknowledge Alarm

|  |  |  |
| --- | --- | --- |
| **void DpDiag\_AlarmAckReq( ALARM\_STATUS\_PDU\_PTR psAlarm )** | | |
| Function | The slave acknowledges an alarm to the user that was set previously: The slave receives the acknowledgement in DPV1 operation from the parameterization master, and tranfers it to the user. | |
| Parameter | psAlarm | Pointer to alarm structure |
| Return Value | None |  |

**Figure 5-51 : Function DpDiag\_AlarmAckReq()**

## Notes:

## Isochron Mode (IsoM)

### General

The IsoM synchronize DP-Master, DP-Slave and DP-Cycle. The isochron cycle time starts with the transmission of the SYNCH telegram by the IsoM Master. If the VPC 3+ supports the IsoM, a **synchronization signal at Pin 13** is generated by reception of a SYNCH telegram.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | Control\_Command |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Group\_Select |

**Figure 6-1 : SYNCH telegram**

There are two operation modes for cyclic synchronization available in VPC3+:

 **Isochron Mode:** Each SYNCH telegram causes an impulse on the SYNC output and a New\_GC\_Command interrupt.

 **Poor Sync:** A Data\_Exchange telegram no longer causes an DX\_Out interrupt immediately, rather the event is stored in a flag. By a following SYNCH message reception, the DX\_Out interrupt and a synchronization signal are generated at the same time. Additionally a New\_GC\_Command interrupt is produced, as the SYNCH telegram behaves like a regular Global\_Control telegram to the DP state machine. If no Data\_Exchange telegram precedes the SYNCH telegram, only the New\_GC\_Command interrupt is generated.

telegrams

IsoM

SYNC

DX\_Out\* New\_GC\_Command\*

Poor Sync

SYNC

DX\_Out\* New\_GC\_Command\*

Data\_Ex SYNCH SYNCH Data\_Ex GC SYNCH





**Figure 6-2 : SYNC-signal and interrupts for synchronization modes**

### Isochron Mode

**Settings for Isochron mode in the DpCfg.h**

The user connects the different services via #define in “DpCfg.h”, so that the program code is adapted to the required services respectively. SYNC\_Ena in Mode Register 2 must be set. Furthermore the polarity (SYNC\_Pol) can be adjusted. Sync\_PW Register contains a multiplicator with base of 1/12 s to adapt the pulse width. Additionally the Spec\_Clear\_Mode in Mode Register 0 must be set.

|  |  |
| --- | --- |
| **Service** |  |
| #define DP\_ISOCHRON\_MODE | Activation of the functionality for the expansion services of the isochron mode. |

**Figure 6-3 : PROFIBUS Services**

|  |  |  |
| --- | --- | --- |
| **Settings for Isochron Mode** |  |  |
| #define SYNCH\_PULSEWIDTH | uint8\_t | Width of Synchpulse in 1/12µs |

**Figure 6-4 : Settings for Isochron Mode**

|  |  |
| --- | --- |
|  | **Mode Register 2, Address 0CH:** |
| bit 7 - 5 |  |
| bit 4 | **SYNC\_Ena:** Enable generation of SYNC pulse (for Isochron Mode only)  0 = SYNC pulse generation is disabled (default).  **1 = SYNC pulse generation is enabled.** |
| bit 3 - 0 |  |

**Figure 6-5 : General Slave Parameter**

|  |  |
| --- | --- |
|  | **Mode Register 0, High-Byte, Address 07H (Intel):** |
| Bit 15 - 14 |  |
| Bit 13 | **Spec\_Clear\_Mode:** Special Clear Mode (Fail Safe Mode)  0 = No special clear mode.  **1 = Special clear mode. VPC3+ will accept data telegrams with data unit = 0** |
| Bit 12 - 8 |  |

**Figure 6-6 : Coding of Mode Register 0, High-Byte**

Settings in Set\_Param telegram are shown below (Master configuration).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 0 |  |  | Sync\_Req  = 0 | Freeze\_Req  = 0 |  |  |  |  | Station\_Status |
| 1 |  |  |  |  |  |  |  |  | WD\_Fact\_1 |
| 2 |  |  |  |  |  |  |  |  | WD\_Fact\_2 |
| 3 |  |  |  |  |  |  |  |  | minTSDR |
| 4 |  |  |  |  |  |  |  |  | Ident\_Number\_High |
| 5 |  |  |  |  |  |  |  |  | Ident\_Number\_Low |
| 6 | Group\_8  = 0 |  |  |  |  |  |  |  | Group\_Ident |
| 7 |  | Fail\_Safe  = 1 |  |  |  |  |  |  | DPV1\_Status\_1 |
| 8 |  |  |  |  |  |  |  |  | DPV1\_Status\_2 |
| 9 |  |  |  | IsoM\_Req  = 1 |  |  |  |  | DPV1\_Status\_3 |
| 10  : 246 |  |  |  |  |  |  |  |  | User\_Prm\_Data |

**Figure 6-7 : Format of Set\_Param for IsoM**

### Poor Sync Mode

**Settings for Poor Sync mode in the DpCfg.h**

DX\_Int\_Port in Mode Register 2 must be set and SYNC\_Ena need not to be set. The setting of polarity and pulse width are the same as by IsoM. Also the Fail Safe Mode must be supported.

|  |  |
| --- | --- |
| **Service** |  |
| #define DP\_ISOCHRON\_MODE | Activation of the functionality for the expansion services of the isochron mode. |

**Figure 6-8 : PROFIBUS Services**

|  |  |  |
| --- | --- | --- |
| **Settings for Isochron Mode** |  |  |
| #define SYNCH\_PULSEWIDTH | uint8\_t | Width of synch pulse in 1/12µs |

**Figure 6-9 : Settings for Isochron Mode**

|  |  |
| --- | --- |
|  | **Mode Register 2, Address 0CH:** |
| bit 7 - 5 |  |
| bit 4 | **SYNC\_Ena:** Enable generation of SYNC pulse (for Isochron Mode only)  **0 = SYNC pulse generation is disabled (default).**  1 = SYNC pulse generation is enabled. |
| bit 3 | **DX\_Int\_Port:** Port mode for Dataexchange Interrupt  0 = DX Interrupt not assigned to port DATA\_EXCH (default).  **1 = DX Interrupt (synchronized to GC-SYNC) assigned to port DATA\_EXCH.** |
| bit 2 - 0 |  |

**Figure 6-10 : General Slave Parameter**

|  |  |
| --- | --- |
|  | **Mode Register 0, High-Byte, Address 07H (Intel):** |
| Bit 15 - 14 |  |
| Bit 13 | **Spec\_Clear\_Mode:** Special Clear Mode (Fail Safe Mode)  0 = No special clear mode.  **1 = Special clear mode. VPC3+ will accept data telegrams with data unit = 0** |
| Bit 12 - 8 |  |

**Figure 6-11 : Coding of Mode Register 0, High-Byte**

Settings in Set\_Param telegram are shown below (Master configuration).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 0 |  |  | Sync\_Req  = 1 | Freeze\_Req  = 1 |  |  |  |  | Station\_Status |
| 1 |  |  |  |  |  |  |  |  | WD\_Fact\_1 |
| 2 |  |  |  |  |  |  |  |  | WD\_Fact\_2 |
| 3 |  |  |  |  |  |  |  |  | minTSDR |
| 4 |  |  |  |  |  |  |  |  | Ident\_Number\_High |
| 5 |  |  |  |  |  |  |  |  | Ident\_Number\_Low |
| 6 | Group\_8  = 1 |  |  |  |  |  |  |  | Group\_Ident |
| 7 |  |  |  |  |  |  |  |  | DPV1\_Status\_1 |
| 8 |  |  |  |  |  |  |  |  | DPV1\_Status\_2 |
| 9 |  |  |  |  |  |  |  |  | DPV1\_Status\_3 |
| 2  : 246 |  |  |  |  |  |  |  |  | User\_Prm\_Data |

**Figure 6-12 : Format of Set\_Prm for DP-Slave using isochrones cycles**

In opposite to IsoM the DX\_Out interrupt first generated by receiving of SYNCH telegram. If no Data\_Exchange telegram received before a SYNCH occurred, no synchronization signal is generated.

### Structured Prm-Data for Isochron Mode

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** |  | **Value range** | **Description** |
| 0 | Structured Length | 28 |  |
| 1 | Structure Type | 4 |  |
| 2 | Slotnumber | 0 |  |
| 3 | Reserved | 0 |  |
| 4 | Version | 1 |  |
| 5 - 8 | TBASE\_DP | 375, 750, 1500 (default),  3000, 6000. All other values are reserved an shall not be used. |  |
| 9 - 10 | TDP | 154 to 216-1 |  |
| 11 | TMAPC | 0 to255 |  |
| 12 - 15 | TBASE\_IO | 375, 750, 1500 (default),  3000, 6000. All other values are reserved an shall not be used. |  |
| 16 - 17 | TI | 0 to 216-1 |  |
| 18 - 19 | TO | 0 to 216-1 |  |
| 20 - 23 | TDX | 0 to 232-1 |  |
| 24 - 25 | TPLL\_W | 1 to 216-1 |  |
| 26 - 27 | TPLL\_D | 0 to 216-1 |  |

**Figure 6-13 : Structured Isochron Mode Parameter**

## Data-eXchange-Broadcast (DXB)

DP-Master (Class1)

Dout

Din

Request (FC=7) Response (DA=127)



Data Exchange with DP-Master (Class 1)

Dout

Din

DP-Slave (Publisher)

Data Exchange with filtered

DP-Master (Class 1) Broadcast (Input) Data

from Publisher

Dout

Din

DXBout

DP-Slave (Subscriber)

Link

**Figure 6-14 : Overview DXB**

The DXB mechanism enables a fast slave-to-slave communication. A slave which holds input data significant for other slaves, works as a Publisher. The Publisher can handle a special kind of Data Exchange request from the master and sends its answer as a broadcast telegram. Other slaves, that are parameterized as Subscribers, can monitor this telegram. A link is opened to the Publisher if the address of the Publisher is registered in the link table of the Subscriber. If the link were established correctly, the Subscriber can fetch the input data from the Publisher. The VPC 3+ can handle a maximum of 29 links.

### Publisher

The VPC3+ handles the publisher mode automatically. In the firmware no adjustments need to be made. A Publisher is activated with 'Publisher\_Enable = 1' in DPV1\_Status\_1. The time minTSDR must be set to 'TID1 = 37 tbit + 2 TSET + TQUI'.

All Data\_Exchange telegrams containing the function code 7 (Send and Request Data Brct) are responded with destination address 127. If Publisher mode is not enabled, these requests are ignored.

### Subscriber

A Subscriber requires information about the links to its Publishers. These settings are contained in a DXB Linktable or DXB Subscribertable and transferred via the Structured\_Prm\_Data in a Set\_Param or Set\_Ext\_Prm telegram. Each Structured\_Prm\_Data is treated like the User\_Prm\_Data and therefore evaluated by the user. From the received data the user must generate DXB\_Link\_Buf and DXB\_Status Buf entries. The watchdog must be enabled to make use of the monitoring mechanism. This must be checked by the user.

### Structured PRM-Data: DXB Linktable

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 0 |  |  |  |  |  |  |  |  | Structured\_Length |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | Structure\_Type |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Slot\_Number |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Reserved |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Version |
| 5 |  |  |  |  |  |  |  |  | Publisher\_Addr |
| 6 |  |  |  |  |  |  |  |  | Publisher\_Length |
| 7 |  |  |  |  |  |  |  |  | Sample\_Offset |
| 8 |  |  |  |  |  |  |  |  | Sample\_Length |
| 9  : 120 |  |  |  |  |  |  |  |  | Further link entries |

**Figure 6-15 : Format of the Structured\_Prm\_Data with DXB-Linktable (specific link is grey scaled)**

### Structured PRM-Data: DXB Subscribertable

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 0 |  |  |  |  |  |  |  |  | Structured\_Length |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | Structure\_Type |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Slot\_Number |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Reserved |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Version |
| 5 |  |  |  |  |  |  |  |  | Publisher\_Addr |
| 6 |  |  |  |  |  |  |  |  | Publisher\_Length |
| 7 |  |  |  |  |  |  |  |  | Sample\_Offset |
| 8 |  |  |  |  |  |  |  |  | Dest\_Slot\_Number |
| 9 |  |  |  |  |  |  |  |  | Offset\_Data\_Area |
| 10 |  |  |  |  |  |  |  |  | Sample\_Length |
| 11  : 120 |  |  |  |  |  |  |  |  | further link entries |

**Figure 6-16: Format of the Structured\_Prm\_Data with DXB-Subscribertable (specific link is grey scaled)**

The user must copy the link entries of DXB-Linktable or DXB- Subscribertable, without Dest\_Slot\_Number and Offset\_Data\_Area, in the DXB\_Link\_Buf and set R\_Len\_DXB\_Link\_Buf. Also the user must enter the default status message in DXB\_Status\_Buf from the DXB-Linktable and write the appropriate values to R\_Len\_DXB\_Status\_Buf. After that, the parameterization interrupt can be acknowledged.

### Structure of VPC3+ DXB-Link Table

|  |  |
| --- | --- |
| **Byte** | **Entry** |
| 0 | Publisher\_Addr (= 0...125) |
| 1 | Publisher\_Length (= 1...244) |
| 2 | Sample\_Offset (= 0...243) |
| 3 | Sample\_Length (= 1..244) |
| ... | ... |
| m - 3 | Publisher\_Addr (= 0..125) |
| m - 2 | Publisher\_Length (= 1..244) |
| m - 1 | Sample\_Offset (= 0..243) |
| m | Sample\_Length (= 1..244) |

**Figure 6-17 : Structure of VPC3+ DXB\_LINK\_TABLE**

### Structure of VPC3+ DXB Link Status

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 0 | 0 | 0 | Block\_Length | | | | | | Header\_Byte |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | Status\_Type |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Slot\_Number |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Status\_Specifier |
| 4 |  |  |  |  |  |  |  |  | Publisher\_Addr |
| 5 | Link\_ Failure | Link\_ Error | 0 | 0 | 0 | 0 | 0 | Data\_ Exist | Link\_Status |
| 6  : 61 |  |  |  |  |  |  |  |  | Further link entries |

|  |  |
| --- | --- |
| **Link\_Status:** | |
| Bit 7 | **Link\_Status :**  1 = active, valid data receipt during last monitoring period  0 = not active, no valid data receipt during last monitoring period (DEFAULT) |
| Bit 6 | **Link\_Error:**  0 = no faulty Broadcast data receipt (DEFAULT)  1 = wrong length, error occurred by reception |
| Bit 0 | **Data\_Exist:**  0 = no correct Broadcast data receipt during current monitoring period (DEFAULT)  1 = error free reception of Broadcast data during current monitoring period |

**Figure 6-18 : DXB\_Link\_Status\_Buf (specific link is grey scaled)**

### Functional Description of the DXB Services

|  |  |  |
| --- | --- | --- |
| **VPC3\_SET\_DXB\_LINK\_TABLE\_LEN (uint8\_t link\_len)** | | |
| Function | Set the length of the DXB-Link Table buffer | |
| Parameter | Length of DXB-Link Table buffer | |
| Return Value | None |  |

**Figure 6-19 : Function VPC3\_SET\_DXB\_LINK\_TABLE\_LEN**

|  |  |  |
| --- | --- | --- |
| **uint8\_t VPC3\_GET\_DXB\_LINK\_TABLE\_LEN ()** | | |
| Function | Get the length of the DXB-Link Table buffer | |
| Parameter | None | |
| Return Value | Length of DXB-Link Table buffer |  |

**Figure 6-20 : Function VPC3\_GET\_DXB\_LINK\_TABLE\_LEN**

|  |  |  |
| --- | --- | --- |
| **VPC3\_UNSIGNED8\_PTR VPC3\_GET\_DXB\_LINK\_TABLE\_BUF\_PTR ()** | | |
| Function | Fetch buffer pointer of the DXB-Link Table buffer. | |
| Parameter | None | |
| Return Value | pointer to the DXB-Link Table buffer |  |

**Figure 6-21 : Function VPC3\_GET\_DXB\_LINK\_BUF\_PTR**

|  |  |  |
| --- | --- | --- |
| **VPC3\_SET\_DXB\_LINK\_STATUS\_LEN (uint8\_t status\_len)** | | |
| Function | Set the length of the DXB-Link Status buffer | |
| Parameter | Length of DXB-Link Status buffer | |
| Return Value | None |  |

**Figure 6-22 : Function VPC3\_SET\_DXB\_LINK\_STATUS\_LEN**

|  |  |  |
| --- | --- | --- |
| **uint8\_t VPC3\_GET\_DXB\_LINK\_STATUS\_LEN ()** | | |
| Function | Get the length of the DXB-Link Status buffer | |
| Parameter | None | |
| Return Value | Length of DXB-Link Status buffer |  |

**Figure 6-23 : Function VPC3\_GET\_DXB\_LINK\_STATUS\_LEN**

|  |  |  |
| --- | --- | --- |
| **VPC3\_UNSIGNED8\_PTR VPC3\_GET\_DXB\_LINK\_STATUS\_BUF\_PTR( void )** | | |
| Function | Fetch buffer pointer of the DXB-Link Status buffer. | |
| Parameter | None | |
| Return Value | pointer to the DXB-Link Status data buffer |  |

**Figure 6-24 : Function VPC3\_GET\_DXB\_LINK\_STATUS\_BUF\_PTR()**

|  |  |  |
| --- | --- | --- |
| **void VPC3\_SubscriberToLinkTable (PRM\_SUBSCRIBER\_TABLE\_PTR psDxb, uint8\_t bNrOfLinks )** | | |
| Function | Converts the dxb-subscriber table format to the dxb-link table format and initialize the VPC3+ with the dxb-link table. | |
| Parameter | psDxb bNrOfLinks | |
| Return Value | None |  |

**Figure 6-25 : Function VPC3\_SubscriberToLinkTable ()**

|  |  |  |
| --- | --- | --- |
| **uint8\_t VPC3\_CheckDxbLinkTable( void )** | | |
| Function | Checks the dxb-link table. | |
| Parameter | None | |
| Return Value | DP\_OK DP\_PRM\_DXB\_ERROR |  |

**Figure 6-26 : Function VPC3\_CheckDxbLinkTable ()**

|  |  |  |
| --- | --- | --- |
| **void VPC3\_BuildDxbLinkStatus( void )** | | |
| Function | Generate from the dxb-link table the dxb link status table and initialize the VPC3+ with the dxb-link status table. | |
| Parameter | Valid DXB-Link Table | |
| Return Value | None |  |

**Figure 6-27 : Function Vpc3\_BuildDxbLinkStatus()**

### Processing Sequence

The VPC 3+ processes DXBout buffers like the Dout buffers. The only difference is, that the DXBout buffers are not cleared by the VPC 3+.

The VPC 3+ writes the received and filtered broadcast data in the DXBout buffer. The buffer contains also the Publisher\_Address and the Sample\_Length.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | **Bit Position** | | | | | | | | **Designation** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 0 |  |  |  |  |  |  |  |  | Publisher\_Addr |
| 1 |  |  |  |  |  |  |  |  | Sample\_Length |
| 2  : 246 |  |  |  |  |  |  |  |  | Sample\_Data |

**Figure 6-28 : Structure of DXBout Buffer**

|  |  |  |
| --- | --- | --- |
| **VPC3\_UNSIGNED8\_PTR VPC3\_GetDxbOutBufPtr ()** | | |
| Function | Fetch buffer pointer of the DXB output buffer. | |
| Parameter | None | |
| Return Value | Pointer to the DXB data buffer  NIL, if no diagnostics buffer in the ‘U’ state |  |

**Figure 6-29 : Function VPC3\_GetDxbOutBufPtr()**

### Monitoring

After receiving the DXB data the Link\_Status in DXB\_Status\_Buf of the concerning Publisher is updated. In case of an error the bit Link\_Error is set. If the processing is finished without errors, the bit Data\_Exist is set.

In state Data\_Exchange the links are monitored in intervals defined by the parameterized watchdog time. After the monitoring time runs out, the VPC 3+ evaluates the Link\_Status of each Publisher and updates the bit Link\_Failure. The timer restarts again automatically.

|  |  |  |  |
| --- | --- | --- | --- |
| **Event** | **Link\_Status** | **Link\_Error** | **Data\_Exist** |
| WD\_Time elapsed AND Data\_Exist = 1 | 0 | 0 | 0 |
| WD\_Time elapsed AND (Data\_Exist = 0 OR Link\_Error = 1) | 1 |  |  |
| faulty DXB data receipt |  | 1 | 0 |
| valid DXB data receipt |  | 0 | 1 |

**Figure 6-30 : Link\_Status handling**

**To enable the monitoring of Publisher-Subscriber links the watchdog timer must be enabled in the Set\_Param telegram. This must be checked by user.**