



用户手册
User Guide

接口卡的编程 Programming with Interface Cards

Elektro-Automatik GmbH



	页面
1. 基本信息	4
1.1 术语解释.....	4
1.2 序言.....	4
1.3 通讯一般注意事项.....	4
1.4 关于USB驱动器.....	4
1.5 通讯结构.....	4
1.6 串行传输的设置.....	4
1.7 设定值与实际值的传输.....	4
1.8 电报结构 USB/RS232.....	5
1.9 CAN卡的消息结构.....	5
1.9.1 带产品节点和RID的旧版CAN系统.....	5
1.9.2 新版CAN系统.....	6
1.9.3 拆分消息.....	6
1.10 IF-G1卡的消息结构.....	6
1.11 IF-Ex (Ethernet)卡的消息结构.....	6
1.12 IF-Ex (USB)卡的消息结构.....	6
1.13 消息定时.....	6
2. LabView的通讯	7
2.1 Labview VIs概览.....	7
2.2 安装.....	7
2.3 VIs概览.....	7
3. 与产品通讯	7
3.1 基本信息.....	7
3.1.1 USB驱动库注意事项.....	7
3.2 标准程序.....	7
3.3 建立电报.....	8
3.3.1 起始分隔符.....	8
3.3.2 掩码.....	8
3.3.3 实例.....	8
3.4 时间格式.....	9
3.4.1 电子负载的时间格式.....	9
3.4.2 电源和电池充电器的时间格式.....	9
3.5 提示.....	10
3.6 问题解答.....	10
3.7 错误消息.....	10
3.7.1 部分错误代码的解释.....	10
3.7.2 通讯错误列表.....	11
3.8 通讯对象清单.....	11
3.8.1 定义列.....	11
3.8.2 对象举例和解释.....	12
3.8.3 关于用户配置文档.....	12
3.9 产品错误、报警代码和报警类型.....	13
4. Profibus-现场总线	14
4.1 基本信息.....	14
4.2 技术规格.....	14
4.3 支持产品系列.....	14
4.4 主-从通讯.....	14
4.4.1 循环.....	14
4.4.2 非循环.....	14
4.5 连线/终端.....	14
4.6 传输速度与延时.....	15
4.6.1 线长对它的影响.....	15
4.7 操作模式.....	15
4.8 现场总线地址.....	15

	页面
4.9 对象描述.....	16
4.9.1 循环式现场总线数据.....	16
4.9.2 非循环式现场总线数据.....	16
4.9.3 数据报的使用.....	17
4.10 以Siemens PLC项目计划为例.....	18
4.10.1 将电源整合到现场总线.....	18
4.10.2 GSD文档的安装.....	18
4.10.3 模块布局.....	18
4.10.4 给循环模块寻址.....	19
4.10.5 给非循环模块寻址.....	19
4.11 诊断.....	19
4.11.1 标准诊断.....	19
4.11.2 DP-V1 报警管理器（扩展诊断）.....	19

注意!

本文档不适用通过 **GPIB卡IF-G1**或**网卡IF-E1/IF-E2**用指令进行的基于文本的通讯! 可参考另外关于这些接口卡的文件!

1. 基本信息

1.1 术语解释

Telegram: 不同长度的字节链。要么发送给产品, 要么从产品发出。

Singlecast: 向单个产品查询简单的信息。如果产品是以链式连接, 像CAN一样, 则所有产品都将收到这个电报, 但是仅由一个地址接收。它与起始分隔符(消息的第一个字节)和产品位址相关。

Broadcast: 向所有产品查询简单信息。如果多台产品像链条一样链接在一起, 比如在CAN系统下, 则链上的所有产品都会收到该电报并接受。利用0产品节点经CAN以外的端口也能访问产品。对于CAN以外的接口, 不管产品的节点设为多少, 广播类型消息都可通过0产品节点(消息的第二个字节)来简化产品的识别。

Object: 就这个词的性质, 它描述的是对象位址, 启动对象产品上定义的反应活动, 与指令类似。

Message: CAN特定数据包, 像无起始分隔符和校验位电报。

Signal: 消息的一部分(该术语用于Vector软件中)

1.2 序言

下面即将描述的面向对象电报结构的通讯协议非常复杂。因此建议使用现成的LabView组件。整合到像Visual Basic、C或.NET的其它环境, 要求掌握CAN或USB硬件接口设置和使用的编程方面的知识, 以及其驱动器的寻址。

我们这儿只解释了数据包(电报)的结构和如何正确传输。

1.3 通讯一般注意事项

给不同型号产品的固件编程, 要尽量考虑多的环境, 包括一次性控制多台产品时出现的环境。因此不可能于任何时候, 在产品的任何状态下执行任何操作。例如: PSI 9000与PSI 8000系列函数管理数据只能在产品待机状态下传输(见产品说明书)。否则将反馈出一错误信息, 提示用户产品不在待机模式。

1.4 关于USB驱动器

在Windows XP, Windows 2003, Windows Vista 以及Windows 7系统下, 该驱动将安装两个设备。其中一个叫“USB串口转换器”的USB设备, 另一个叫“USB串行端口”的虚拟COM串口(VCP)。

驱动器的VCP功能默认被激活, 并将为每个连接到电脑上的USB端口创建一个新的虚拟COM端口。

目前的LabView VIs软件仅经该虚拟COM端口利用IF-Ux, IF-Ex或IF-PB1卡的USB端口就能与产品通讯。除此之外, VIs软件还支持IF-Ex卡和GPIB (IF-G1)卡的以太网端口。可从USB芯片制造商的这个网站www.ftdichip.com, 下载比光碟上驱动程序更新的版本。

1.5 通讯结构

用控制器通讯基于下面这些电报类型:

a) 消息: 发送的对象可以是设定输出电压(举例来说)。只要产品当前状态允许此动作, 即接受和执行对象。产品不会发送任何回答。如果不允许, 则以一错误消息发送回复。

b) 查询: 通过使用某特定对象发送查询, 如“获得实际值”, 则期望得到一回答。如果产品当前状态允许该查询, 即执行并回答。该回答包含查询的数据。如不允许将发送一错误消息作为回答。

1.6 串行传输的设定

对象: 使用时的RS232和USB端口(也可见1.4章节)

利用一个字节串行传输可发送下列字节:

开始位+8个数据位+奇偶位+停止位

奇偶性被检测为“奇数”。

IF-Ux, IF-Ex和IF-PB1的USB端口内部以RS232卡的特性工作。要求至少为这类卡的特定驱动器设定通讯参数, 使用RS232卡时也一样:

RS232卡的波特率: 9600Bd ... 57600Bd

RS232卡的波特率: 56700Bd

奇偶性: 奇数

停止位: 1

1.7 设定值与实际值的传输

开设定值和实际值(见另外对象清单)是少数几个特例, 它们以百分比值传输的。即0x6400对应100,00%。故传输前的任何实际设定值, 以及传输后收到的任何实际值都需转化。

如果产品额定电压为80V, 查询的实际值为0x3200 (0x3200 = 50,00%), 则它对应的是40V输出电压。

高字节为百分比数字(0x64 = 百分比值小数点前数字), 而低字节则为百分比值小数点后数字。

$$\text{实际值} = \frac{\text{额定值} * \text{百分比值}}{25600}$$

举例: 如果产品最大值为80V, 进来的百分比实际值为0x2454 = 9300。于是得到: 实际值 = (80 * 9300) / 25600 = 29.06V

$$\text{要发送值} = \frac{25600 * \text{实际值}}{\text{额定值}}$$

举例: 如果设定功率为500W, 产品最大电压为640W。按照公式将得到:

$$\text{百分比设定值} = (25600 * 500) / 640 = 20000 = 0x4E20$$

1.8 电报结构 USB/RS232

IF-Rx (RS232)接口卡, 跟IF-Ux, IF-Ex (USB)与IF-U1接口卡的端口使用一种与CAN卡IF-Cx略微不同的电报结构。如果您正在使用CAN卡, 请跳到章节1.9。

电报结构类似这个

SD + DN + OBJ + DATEN + CS

并由这些字节组成:

Byte 0: SD (start delimiter-起始分隔符)

起始分隔符决定如何操作电报。字节解释如下:

Bits 3-0: 数据长度 (3-18个字节)

这些字节定义电报数据下-1数据长度。查询时, 在此给出期望数据长度。

举例: [对象清单](#)描述了某一对象的6数据长度, 故SD较低的半字节将为5。

例外: “字符串”型对象无关乎数据长度, 只要它不高于对象清单给出的数值, 因此0与对象长度间的任何值都可设定。

Bit 4:

0 = 产品给电脑发电报

1 = 电脑给产品发电报

Bit 5:

0 = 单播, 电报传送给某一产品节点

1 = 广播, 电报传送给所有产品节点 (用CAN卡), 否则跟单播一样, 利用DN = 0

Bits 6+7: 传输类型

00= 保留

01= 查询数据

10= 回答查询

11= 发送数据

Byte 1: DN (device node--产品节点)

产品节点在总线系统 (如CAN或GPIB) 中识别产品和寻找产品地址。每个节点数仅能分配一次。因为用它来寻找一特定产品的地址。数值范围为: 1...30, 其它数无效。使用CAN时, CAN ID是根据产品节点计算出来的 (也根据产品型号和固件版本而定)。详情见章节1.9。在点对点系统下 (如RS232或USB), 当使用“单播”型消息时位址才很重要。使用广播时, DN可一直为0。

Byte 2: OBJ

用此字节指向产品通讯对象的地址。在通讯[对象清单](#)内, 详细解释了对对象及其功能。

Byte 3 - 18: Data field

数据区可能有1-16个字节长, 因此电报的长度也会随之变化。如果发送 (电脑->产品) 查询, 将无数据 (=数据区为空), 随字节2之后直接就是电报校验码 (见下面)。只有在发送一回复 (产品->电脑), 一期望回复或一错误, 都会有一特定长度的数据。

Last 2 bytes: CS (check sum--校验码)

校验码始终在电报的末尾。它由电报前面的所有字节简单相加而成, 并附于尾部。它为两个字节长 (16个字节值)。高字节位于低字节前面。

电报范例:

可用产品节点1将对象号71 (查询实际值) 发给产品。电报必须像如下的十六进制数值一样:

55 01 47 00 9D

回复可能为:

85 01 47 64 00 1E 00 50 00 01 9F

(这翻译出来就是一台80V, 100A和3000W的电源实际输出80V, 30A和2400W, 举例如PSI9080-100。)

也可见章节1.7, 关于设定值和实际值的转换。

章节3.4和3.8.2有更多范例。

1.9 CAN卡的消息结构



注意!

PS 8000 (自6.01固件版本开始), **PSI 8000** (自3.14固件版本开始) 与 **EL 3000/EL 9000** (自5.01固件版本开始) 开始有新的CAN ID系统, 如下述。其他系列仅支持1.9.1系列章节所述的系统。

重要信息! 自2011年6月

IF-C1接口卡支持CAN V2.0a标准。不适用扩展地址格式。

关于产品的使用说明与CAN接口卡的具体通讯设定。

1.9.1 带产品节点和RID的旧版CAN系统

注意: CAN系统可应用到支持IF-Cx CAN接口卡的每个系列产品上, 除固件版为6.01以及更高的PS 8000系列产品。

CAN启动芯片需要标识符, 传输最多8个数据字节和数据长度。标识符达11位字节长 (CAN 2.0a), 用产品字节、可重定位标识符段(RID)和消息类型来表示。每个产品我们定义两个标识符:

[RID*64 + device node * 2] (第一个标识符) 和

[RID*64 + device node * 2 + 1] (第二个标识符),

但是第一个标识符发送消息给产品以设定某参数, 第二个发送消息以查询某参数和接收回复。

一个消息可以包含最多8个字节。第一个字节为通讯对象码。之后您可加上其它7个数据字节 (见通讯[对象清单](#))。

用16字节长的数据段发送对象, 需要发送至少三个消息, 数据段被分成这三个消息。见章节1.9.3描述的更多详情。

两个范例:

a) 产品将要设为远程模式。这需要用状态指令或设定值控制产品。device node设为15, RID为3。消息为“仅发送”类型。发送的标识符按 $3 * 64 + 15 * 2 = 222$ 计算, 或按上述0xDE格式计算。根据[对象清单](#), 我们可以使用数据字节0x10 (掩码) 和0x10 (设定远程模式) 的对象54 (十六进制: 0x36)。从而形成的数据长度为3, 故CAN数据形成如下:

ID DL DATA

DE 03 36 10 10

b) 如果不想设置状态, 但要查询它, 可使用0xDF标识符。(第二个标识符), 因为它为查询, 对象号足以作为数据。故CAN数据如下:

DF 01 36

回复应该如下:

DF 01 36 10 10

1.9.2 新版CAN系统

注意：PS 8000系列产品自2011年6月开始进行的固件更新支持CAN ID系统。其他系列陆续会支持。更新后的固件含有一新的CAN ID系统，旧系统不再支持，指定的电脑软件可能会停止工作。

新的CAN ID系统使用3个基本ID，外加一个通讯用广播ID（每台机）。它与Vector软件兼容，如CANalyzer或CANoe。按客户需求可提供*.dbc格式的Vector数据库，也可刻录在光盘上与CAN接口卡一起提供。在这些软件内还可打开即时使用的配置做演示用。

若配有IF-C1或IF-C2 CAN接口卡，在产品设置菜单下，用户调出一个基本ID，然后产品使用三个CAN ID（基本ID，基本ID + 1和基本ID + 2）。在系统总线上这些ID必须是唯一的，这样才可在第四步完成基本ID的调节。如果总线上连接的是多台同型号产品，每台都需使用不同的基本ID。

关于广播ID，则与之相反。对于多台产品建议调节成相同的广播ID。目的在于给总线上的x台产品发送相同的信息，从而在同一时间设置相同的设定电压。在第一步将广播ID与基本ID分开调节，且必须是唯一的。意思是，必须与产品上其它CAN ID不同。

三个基本CAN ID按下面格式进行分配：

基本 ID = 电脑仅用来发送设定数值或条件的信息，且不需创建回复电报

基本 ID + 1 = 电脑通过发送询问对象（见对象的文件（即：指令））询问产品信息

基本 ID + 2 = 远程设备用来发送对询问的回复，或者发送错误信息

广播ID分配如下：

广播 ID = 电脑仅用来给多台产品发送信息，从而设定数值或条件，不需创建回复电报

1.9.3 拆分消息

一个拆分的消息是被拆分成多个子消息的消息（仅针对“字符串”格式的对象）。在对象号（=对象地址）后面插入一个标识符。第一个消息的标识符为0xFF，第二个为0xFE，第三个为0xFD。不指定这些消息的顺序。但是必须用这些拆分后的信息再次组成电报。使用网关功能时，不通过网关组成拆分电报，而用更高级的控制器来完成。

1.10 IF-G1卡的消息结构

经GPIB卡进行基于文本的通讯，其消息结构详细描述请参考另外的用户手册。

链接：[电源用SCPI指令清单](#)

链接：[电子负载用SCPI指令清单](#)

1.11 IF-Ex (Ethernet)卡的消息结构

经IF-Ex卡的以太网端口进行基于文本的通讯，其消息结构详细描述请参考另外的用户手册。

链接：[电源用SCPI指令清单](#)

链接：[电子负载用SCPI指令清单](#)

1.12 IF-Ex (USB)卡的消息结构

请见章节1.6和1.8关于LabView的通讯。

1.13 消息定时

每次查询后，产品需要5ms至最长50ms的时间回答。基本上可直接一个接着一个地发送查询。但是如果收到一事件，须至少等待50ms。不过建议等待100ms，以不让产品的操作由于过度繁重的通讯而慢下来。

使用网关功能时（仅针对PSI 9000系列），需要考虑将电报从一个总线系统传输到另一个所耗费的时间。回复可能延迟最多200ms。

收到网关的错误消息后，在下次传输前至少等待100ms。

CAN卡的广播：

每次广播查询后，所有总线共享的产品仅可一个接着一个地回复。根据总线系统，产品的波特率和数量，以及额外的总线流量，或多或少会延迟回复。时间不可指定，仅能通过总线共享产品数量*单播时的反应时间来估计。多数情况下，反应时间会更短。

2. LabView的通讯

2.1 Labview VIs概览

为使多台且不同型号的产品简单整合到现有Labview应用中，我们提供了一组Labview VIs。

利用这些虚拟仪器（VI），用户无须学习更底层的通讯也可以简单地执行和编程应用。

要使用LabView VIs的功能，需利用和操作National Instruments公司的软件开发工具。LabView VIs支持7.0或更高版本。

最少需要下列系统：

- 奔腾3 CPU，256MB内存
- Windows操作系统(Win98和WinXP)

可从我公司网站下载最新版VIs，或发邮件给我们申请获取。

2.2 安装

在用户的LabView环境下安装和使用VIs，需阅读随附CD上的“installation_english.pdf”文档内的操作说明。

安装完后，在LabView IDE in “Instrument I/O -> Instrument drivers -> IF-XX”的上下文菜单下可找到VIs。

有些VIs只适合特定系列产品，也只能在这类产品上方可工作。这些文件位于安装的子文件夹内。其它可公用。且以PSI9缩写为文件名和VI图标名的前部分。VIs说明书中描述了其用途和功能。可通过LabView上下文帮助(Ctrl+H)以通常方式进入，或直接打开CHM。根据Windows操作系统版本，必须将帮助文件复制到本地硬盘，方便正确阅读。

提示

请在产品上与LabView版本上使用合适的VIs软件版本。
请阅读随附CD上的LabView VIs帮助文件，以了解其操作。

2.3 VIs概览

日期：08/2012

VI套件解压缩后，会有下面几个子文件夹：

“\Common”文件夹下通用的VIs：

- **Device_close.vi** - 关闭特定产品的通讯。退出应用前，若不在使用通讯，应关闭任何已开通讯端口。
- **Device_scan.vi** - 扫描兼容设备的硬件端口（RS232，GPIB，Ethernet，VCP），并返回已寻设备的号码，以及其它VIs需要的产品信息清单。
- **Device_select.vi** - 通用于选择和打开特定产品上由Device_scan.vi返回的清单。也可选择多台产品，并并联使用。

文件夹下特定系列产品的VIs：

- „\BCI8 Series\” - 适用BCI 800 R系列所有型号
- „\PS2 Series\” - 适用PS 2000 B系列所有型号
- „\PS 8 Series“ - 适用PS 8000系列所有型号
- „\PSI 8 Series“ - 适用PSI 8000和PSI 800 R系列所有型号
- „\PSI 9 Series“ - 适用PSI 9000系列所有型号
- „\EL Series“ - 适用EL 3000和EL 9000系列所有型号

„\Examples“文件夹下的实例为了演示通用和特定VIs的用途和置放顺序。实例中的特定VIs也可用其他VIs代替，以测试不同产品类型。



注意！

不管何时用哪一个产品的特定VI代替另一个产品的VI，必须删除“command list”输入脚的枚举常数，然后重新创建！

3. 与产品通讯

3.1 基本信息

下面这一章节描述通讯电报的构成，产品状态的通讯独立性，和与该主题相关的问题，没有关于在低级别下使用USB端口时如何运行USB驱动程序的解释，也没有关于如何正确发送CAN消息的解释。这些都需用户自己学习和完成。

3.1.1 USB驱动库注意事项

使用USB卡IF-Ux，或以网卡IF-Ex，或现场总线卡IF-PB1的USB端口时，需安装USB驱动程序。安装后，用户可选择USB驱动程序进行低级别的访问，或通过虚拟COM端口(VCP)通访问VCP更简单。

随附产品的CD上，在文件夹\manuals\other\ftdi下，有一个PDF文档，详细描述了USB驱动程序DLL的功能。一般情况下，先打开（FT_Open或类似）产品（现在是USB硬件），然后配置（FT_SetBaudRate, FT_SetDataCharacteristics等），并写（FT_Write）或读（FT_GetQueueStatus, FT_Read）。只要不再使用产品，既建议关闭（FT_Close），但也不是每完成一次读-取循环都建议打开和关闭。USB硬件的配置需在通电后一次性完成。FT_Write和FT_Read用来传输下节描述的面向对象通讯的实际电报字节。

3.2 标准程序

只要利用接口卡进行产品的编程，总是遵循相同的步骤。只是特定产品系列支持的通讯对象号码和功能有所不同而已：

一般应用如下：

- 仅可进行监控，即：查询实际值和状态。产品不需要进入远程模式。
- 产品状态和设定值的设定被看作是控制，需要激活远程模式（这时的远程指经数字接口卡遥控产品）。
- 可通过某个条件阻止远程模式。例如：产品明确显示处于本地操作模式（只有PSI 9000），或不同的模式且不允许远程控制。关于更多详情请参考产品说明书。

开始控制产品时，如：发送一设定值，您需要：

1. 激活遥控模式（对象54）

2. 发送设定值

而且，若尚未完成：

3. 设置输出/输入为打开状态

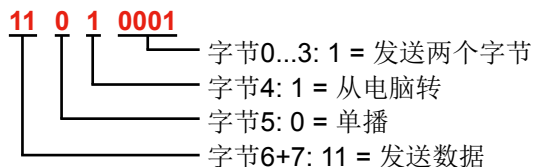
如果不再继续使用，应该离开远程模式。只要远程还是激活状态，就不能手动操作产品，或只能限制性地操作。模式显示于显示器上。

3.3 建立电报

3.3.1 起始分隔符

注意：起始分隔符在CAN卡下不使用

根据电报格式（也见1.8章节），第一个字节为基于电报类型和方向的起始分隔符（SD）。例如：SD即是0xD1，以单字节就是这样：



与字节组不同的还可用十六进制值进行简化。从字节6+7开始，我们可以得到：

SD = 消息类型+播放类型+方向+长度

消息类型为

0xC0 发送数据或

0x40 查询数据

播放类型为

0x00 单播或

0x20 广播

方向为

0x10 从电脑到产品或

0x00 从产品到电脑

数据长度-1 可以是

0x00...0x0F 直至16字节的数据（关于CAN，见章节„1.9.3 拆分消息“）

提示

始终注意要像字节号码-1一样定义数据长度！！

3.3.2 掩码

有些对象要求有一掩码，在控制字节前发送。[对象清单](#)的第7列有列出可能的掩码值。控制字节的字节有多种功能，故需确定那个字节需要转换。一般用掩码1备确定所有要用控制字节更改的字节。举例：字节4要变为1。根据字节值，控制字节将为0x10，掩码为0x10。如果已知一掩码为0x0F，则一直要使用0x0F。然后可一次性更改所有相关字节，举例如对象56（PSI 8000和PSI 9000的函数管理器）。

注意!

为避免这些位元赋予的功能发生逻辑性冲突，建议每个电报只设一个位元！

3.3.3 实例

例 1：将产品转至远程模式

连接设备的地址（节点）设为5，使用54对象（以十六进制表示为0x36），远程模式的掩码（也可见[对象清单](#)）为0x10，控制字节也为0x10。于是我们获得这样一个电报：

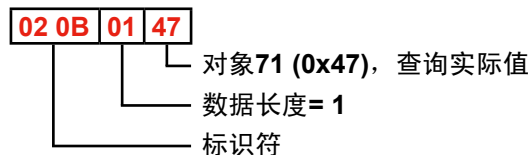
D1 05 36 10 10 01 2C

要取消这个命令，即取消远程模式，需发送**D1 05 36 10 00 01 1C**。掩码保持不变，仅控制字节改变。

例2：通过CAN卡查询实际值

使用CAN时，不使用起始分隔符SD和用户计算的校验码CS。我们只需要对象(按照[对象清单](#)应为71，hex0x47)，识别ID和字节长度。在一个CAN消息内，对象包含在数据长度里，因此该消息要含有1的数据长度，因为我们只发送查询实际值的对象。若使用旧CAN系统（见1.9.1和1.9.2章节），产品地址（节点）为5，RID为8。根据1.7章节的公式，标识符这样计算： $8 * 64 + 5 * 2 + 1 = 523$ (hex = 0x20B)。+1为“查询”消息类型。

现在发送一个字节到ID 0x20B。CAN消息必须像这样（根据Windows GUI上显示的）：



注意!

这不是实际发送到CAN总线的CAN消息字节组合。CAN控制器将多个字节合并到一起，然后增加一个自身校验码。这些只是发送到CAN驱动程序的字节。

对这种查询的回答类似如下：

02 0B 06 64 00 0A 00 42 AA

相同的标识符，数据长度为6，因为每次发送的是16位的三个实际值。这些实际值以百分值传输，并转化为实际值，详情请见章节„1.7 设定值与实际值的传输“。对于EL 9080-200产品，其实际值可转化为100%的电压(=80V)，10%的电流(=20A)和66,7%的功率(=1600W)。

注意：

提示

额定功率、电流和电压可通过合适的对象从产品上读取，并将实际值转化为真实值。

3.4 时间格式

时间格式通过一个16位值表示从1 μ s至100h的时间。这个时间标记发送给产品作正确性检查。太高或太低的值不接受，并返回一错误消息。上面4位被当作掩码，决定时间范围，剩下的代表时间值。

时间格式用来写（即设定）或读出时间值。

适用于任何具有与时间功能相关的产品，只要可设定时间值。下表的时间范围分辨率不一定符合特定产品的分辨率。因此要将数值四舍五入。举例：发送0x23E7的时间值，它代表999 x 1 μ s = 999 μ s。在该范围内可手动调节的时间值为0.95ms或1ms。999 μ s四舍五入成为950 μ s。因此回读时将会是0x23B6(=950)，而不是0x23E7。

提示

不是所有的产品使用下表的所有掩码。

3.4.1 电子负载的时间格式

对于电子负载和其上升时间（对象92）适用下表：

时间范围		产品步宽	掩码	数值范围**	
从	至			从	至
30 μ s	99 μ s	1 μ s	0x2000	0x201E	0x2063
0.10ms	0.99ms	10 μ s	0x2000	0x2064	0x23DE
1.0ms	9.9ms	100 μ s	0x3000	0x3064	0x33DE
10ms	99ms	1ms	0x6000	0x6064	0x63DE
100ms	200ms	1ms	0x7000	0x7064	0x70C8

** 不同步宽的值要作四舍五入

对于电子负载和其脉宽（对象90和91）适用下表：

时间范围		产品步宽	掩码	数值范围**	
从	至			从	至
0.05ms	0.95ms	50 μ s	0x2000	0x2032	0x23B6
1.00ms	9.95ms	50 μ s	0x3000	0x3064	0x33E3
10ms	99.9ms	100 μ s	0x6000	0x6064	0x63E7
100ms	999ms	1ms	0x7000	0x7064	0x73E7
1.00s	9.99s	10ms	0x4000	0x4064	0x43E7
10.0s	100s	100ms	0x9000	0x9064	0x93E8

** 不同于步宽的值要作四舍五入

对于电子负载和其**电池测试累积用时(对象64)**适用下表，基于上面的大表：

时间范围		产品步宽	掩码	数值范围**	
从	至			从	至
1s	3599s	1s	0x8000	0x8001	0x8E0F
1h	99h:59m	1m*	0xC000	0xC03C	0xD76F

* 上面的1h这个时间，只能以HH:MM格式读出。

** 不同于步宽的值要作四舍五入

例1：电子负载的上升时间可设为75ms。75ms对应负载的时间范围步宽为1ms。因此我们需使用0x6000的时间范围。其分辨率为0.1ms，故形成750的时间值（75ms : 0.1ms）。它转化为0x2EE。与掩码一起即可获得0x62EE的上升时间（对象92）。

注意：LabView用户需以不同方法提供版本号，见VI文件。

例2：读出电池测试时间，将其转化为普通时间格式。整个电池测试时间的分辨率为1s。由于时间范围仅允许1s的分辨率，超过1h的时间以分钟和小时显示。举例：如果时间值为0x8743，会转化为1859s或30m59s，或、者1330m或22h10m。在这个时间范围秒钟被忽略，因此要在一分钟之内读出同一时间值。

例3：将A（对象90）的脉宽设为5s。根据上面第二个表，时间范围掩码为0x4000。与分辨率结合在一起，可形成以0x1F4十六进制表示的一个500（5s : 0.01s）的数值，然后最终形成的时间值将为0x41F4。

3.4.2 电源和电池充电器的时间格式

适用于：PSI 8000 / PSI 9000 / BCI 800 R

PSI 8000 / PSI 9000系列函数管理器的序列时间值以2ms或它的倍数间隔为时间格式。其它时间值也使用这样的时间格式，像对象39-43和47。

时间范围		产品步宽	掩码	数值范围**	
从	至			从	至
0.002s	9.998s	2ms	0x0000	0x0001	0x1387
10.00s	59.99s	10ms	0x4000	0x43E8	0x576F
1:00m	59:59m	1s	0x8000	0x803C	0x8E0F
1:00h	99:59h	1m	0xC000	0xC03C	0xD733

提示

不是所有从产品上读取的时间格式都需按时间格式规格创建。见**对象清单**。

掩码 *	时间值 (位数 11..0)				分辨率	形成的时间范围
位数15..13 ⁽¹⁾ 或 15..12 ⁽²⁾	最小(十进制)	最小(十六进制)	最大(十进制)	最大(十六进制)		
0x2000 ⁽¹⁾	0	0x00	999	0x3E7	1 μ s	0 ... 0,999ms
0x3000 ⁽²⁾	100	0x64	999	0x3E7	10 μ s	1ms ... 9,99ms
0x6000 ⁽¹⁾	100	0x64	999	0x3E7	100 μ s	10ms ... 99,9ms
0x7000 ⁽²⁾	100	0x64	999	0x3E7	1ms	100ms ... 999ms
0x0000 ⁽¹⁾	0	0x00	4999	0x1387	2ms	0 ... 9,998s
0x4000 ⁽¹⁾	100	0x64	5999	0x176F	10ms	1,00s ... 59,99s
0x8000 ⁽¹⁾	1	0x01	3599	0xE0F	1s	1s ... 59min:59s
0x9000 ⁽²⁾	100	0x64	1000	0x3E8	100ms	10,0s ... 100,0s
0xC000 ⁽¹⁾	0	0x00	5999	0x176F	1m	01:00h ... 99:59h

表：时间格式

* 如果用掩码将时间值转化为实际时间，根据使用的时间范围，或者与15...13相关，或者与15..12相关。

3.5 提示

I. 检测一产品节点（不用GPIB）

举例：如果您想通过USB控制一产品，但又不知道产品节点，您就可使用，比如：0广播节点，然后查询产品级别。产品会以早就设于产品上的自身节点回答。接着就用它来控制 and 区分产品。

使用USB或RS232接口为点对点的连接，这儿可利用0产品节点使用广播消息类型（见章节„3.3.1 起始分隔符“）

II. 远程和待机

对象54用来启动/停用远程控制操作或产品的输入/输出。也可用它来启动这两个功能，但是强烈建议不要这样操作，因为设定输入/输出要求先激活远程控制，否则会产生错误消息。应该先用控制字节内相应的位组激活远程控制，然后以不同控制字节再次发送对象54。停止远程控制也很简单，方法与之相反。

用对象70回读产品状态也非常有用，可以用来检查对象是否已正确设置。

3.6 问题解答

问题：产品对指令无反应或不反馈

可能出在USB上的原因

- USB卡需要驱动程序。检查驱动程序是否正确安装，在设备管理器的“USB controllers”中是否可找到类似于“USB Serial Converter”的卡。
- 使用了错误的产品节点（=地址）进行通讯。

可能出在RS232上的原因

- 您未使用RS232卡的1:1连线。
- 使用了错误的产品节点（=地址）进行通讯。
- 产品和电脑配置的波特率等不同。
- 对应配置的波特率，通讯线太长（见接口卡用户手册的章节“2 技术规格”）

可能出在GPIB上的原因

- 如果有多个产品连接在一个IEEE总线上，可能有一个或多个产品地址重复了。
- 用了错误的语法。举例：负载对指令OUTP不反应，因为它只有输入。或者该指令对您试图连接的产品类型无效。

可能出在CAN上的原因

- 使用了错误的CAN ID。
- 设定了错误的波特率
- 选择了错误的取样点（仅针对PSI 9000系列，见说明书）
- 产品位于总线末端，且未设定终端。

问题：发送了多个查询，但未得到所有回答。

原因：后面发送的查询太快。根据通讯类型和速度，以及执行时间，在两个查询之间应加上一定的反应时间。

经验法则：反应时间=传输时间+执行时间

如果只有一个查询或还需设置一些参数，执行时间一般定为5-20ms。然后根据波特率和传送的位码，可计算出传输时间。

问题：未设置设定值和状态

可能产生的原因：

- 连接设备不在远程控制模式下，或当前不能设为远程控制模式，因为此时不允许设置，或将产品设为远程控制的其它条件未满足。
- 发送的数值错误（太高，太低），或标准值范围（(0...0x6400的电压、电流等）被限制值（仅针对PSI 9000/PSI8000）限定。此时将发出一错误消息。

3.7 错误消息

3.7.2章节下的表格列出了产品返回一错误信息时所有可能出现的错误代码。该代码向用户指出错误的根源。有些错误是因为错误的消息（即：错误校验码等）查询/设置造成的，有些可能是产品发出的，拒绝不同于当前状态的指令。

错误消息以电报格式形成，即他们由一个起始定界符（除CAN外）、对象号（识别错误，0xFF当对象号使用）和数据区内的错误代码与校验码（除CAN外）组成。

举例：若想用对象设置电压，且产品不在远程控制模式，您将从带节点7的产品收到C0 07 FF 09 01 CF错误消息。

3.7.1 部分错误代码的解释

代码0x7：产品未知该电报的对象号。因为不是所有产品类型能够使用相同的对象号。请参考对象清单对照产品。

代码0x8：该电报的数据区长度已在对象清单内定义了。如果发送“int”类型的，总是2个字节的设定值，但是数据区仅含一个字节，则会出现这个错误代码。即使开始定界符含有正确的电报长度，这是对设定错误值的保护。

代码0x9：对象已发出以便设定数值，但是产品不在远程控制模式。在此状态下只允许读，不能写。必须先将产品设至远程控制模式方可。

代码0xE：使用CAN时应以不同方式传递字符串。如果字符串长度超过8个符号，必须用指定的起始符0xFF，0xFE等，拆分消息。见章节„1.9.3 拆分消息“。

代码0xB/0xD/0x14：如果消息被正确执行，将从消息缓冲区中删除。如果消息进入时太快，缓冲区会溢流，任何消息都不能接收。

代码0x30/0x31：这与设定值有关。所有设定值都有一个能在电源上定义的上限和下限。设定值默认上限，假如当前设定值为0x6400（=100%），下限为0。限定值也适用时间值。

代码0x32：发送了使用错误范围的时间值。该值未超过上限或下限，但仍会造成此错误。

代码0x36/0x37：访问这些数据的条件未满足。见对象清单第4栏，关于访问条件。

3.7.2 通讯错误列表

错误代码		
十六进制	小数	描述
01	1	RS232: 奇偶性错误
02	2	RS232: 结构不正确 (起始位或结束位错误)
03	3	校验和不正确
04	4	起始分隔符不正确
05	5	CAN: 超过最大节点
06	6	节点未知/未出现网关
07	7	对象未定义
08	8	对象长度不正确
09	9	违反读/写权限, 不可访问
0A	10	两字节间间隔时间太长 / 信息字节数量错误
0C	12	CAN: 拆分消息终止
0F	15	产品位于“Local-本地”或模拟远程控制模式
10	16	CAN驱动芯片: 写入错误
11	17	CAN驱动芯片: CRC和错误
12	18	CAN驱动芯片: 格式错误
13	19	CAN: 期望数据长度不正确
14	20	CAN驱动芯片: 缓冲区已满
20	32	网关: CAN写入错误
21	33	网关: CAN CRC检查错误
22	34	网关: CAN格式错误
30	48	超过对象上限
31	49	超过对象下限
32	50	未遵守时间定义
33	51	仅可在待机状态下进入菜单参数
36	54	拒绝进入函数数据
38	55	不可访问对象

图解




	通讯错误
	用户错误
	内部错误

表: 通信错误

3.8 通讯对象清单

重点! 产品的对象清单在另一个PDF文档内, 应与该文档一起分发。

链接: [PSI 8000 T / DT / 2U / 3U系列对象清单](#)

链接: [PSI 9000系列对象清单](#)

链接: [PSI 800 R系列对象清单](#)

链接: [BCI 800 R系列对象清单](#)

链接: [EL 3000 和 EL 9000系列对象清单](#)

链接: [PS 8000 T / DT / 2U / 2U系列对象清单](#)

这些对象清单是自定义应用编程的一个参考, 从而可以远程控制产品, 且除LabView之外, 还有可即时使用的LabView VIs。想重建二进制电报通讯的LabView用户也可拿这些清单作参考。

3.8.1 定义列

第1列包含十进制值的对象号码 (=指令)。这个号码必须分配给电报的OBJ字节。

第3列定义对象是否只读 (ro), 即: 仅能向产品查询, 或是否也可写 (w, r/w)。读是一直都可以的, 也称监控。设定数值或状态要求启用远程控制模式 (如果产品不在“local”模式或类似模式)。也见章节3.2。

第4列定义访问对象的特殊条件。这些对象的执行取决于下列条件之一。如果未满足该条件, 则不执行对象, 产品会返回含错误代码的错误消息。号码含义如下: **1** = 必须关闭产品输出/输入

(只有当输出/输入在待机模式下产品才接受和执行对象)

2 = “Internal resistance” 选项必须解锁*

(只有当“Internal resistance”选项必须解锁后, 产品才接受和执行对象)

3 = 启用函数数据的传输*

(只有当用另外一个对象发出指示并设置函数数据之前, 产品才接受和执行对象)

4 = 激活函数管理器

(只有当函数管理器在产品菜单下手动激活或用不同对象激活, 才接受和执行对象)

5 = 函数管理器未激活*

(只有当函数管理器未被激活才接受和执行对象)

**注意!**

一般在发送可改变产品数值的对象前, 要求先将产品设为远程控制模式。

第5列在电报数据区定义数据类型, 一般使用已知数据类型。

第6列定义对象数据的数据长度。带“字符串”数据类型的对象定义字符串的最大长度。字符串用EOL (行尾, =0) 来结尾或以给出字节号为结尾。如使用CAN, 字符串按多达三个拆分电报传输进来。也见„1.9 CAN卡的消息结构“。

第7列用来屏蔽“char”型数据。这个掩码定义即将设置或不设置哪个位元。也可见章节„3.3.2 掩码“。

第8 & 9列详细解释了数据区的内容。

有些对象使用两字节时间格式, 详细解释见章节„3.4 时间格式“。

3.8.2 对象举例和解释

- 可在章节3.8.1中找到对象清单各列的描述。
- 所有号码若未以0x开头表示为十六进制，则都为二进制。

I. 函数管理器（对象56, 58, 73-75, 78, 90-146）

使用函数管理器时，对象的使用顺序变得非常重要。因为设置和控制函数管理器非常复杂，不在此处操作，而是在一外部应用备注（AN001）下操作，该文档可在随附CD的“\manuals\application notes”文件夹中或我公司网站上找到。

II. 对象54

该对象的主要功能可将产品转为远程控制操作，或打开或关闭电源输入/输出。必须已知即将要更改位元的掩码。

举例：激活远程控制->位元4->位元4的值=0x10->掩码0x10->控制字节也为0x10。对象0x32将包含数据0x1010。停用远程控制的方法相同：掩码0x10->控制字节0x00->数据0x1000。

阅读对象需一掩码，也会返回一主掩码，但是仅为信息值。

III. 对象56

这里的位元只允许单独设定。否则会执行不当。

IV. 对象73

时间戳只在使用函数管理器时才出现，否则为0。它代表以2ms的步宽作为运行时间的计数器值。因为这是一个整数，计数器在65536 x 2ms 后，从0重新开始。

V. 对象77

读取报警信息后即清空缓冲器（针对EL 3000或PS 8000）。其它产品系列，如PSI 9000, PSI 8000, PSI 800 R和BCI 800 R则需用对象54上的一个位元进行初始化，以刷新缓冲区。因为它仅存储3个事件，即报警。一般保留头三个事件，然后在“Last alarm type”位置，后面的事件总是覆盖最近的一个。

举例：对象在数据区（最后记录的报警）头两个字节返回0x0120->错误类型0x01意味着，错误仍然存在，错误代码0x20表示：PSI 9000多相电源的上功率级出现过温（见3.9章节下的表）。

VI. 对象39-47

这些对象与PSI 9000或PSI 8000用户说明书的7.6章节有关。用监控值触发的事件，根据对象44-46的配置，在报警缓冲区产生报警、警告或通知类错误（见PSI 9000用户说明书的定义）。

此处的这个时间是事件触发器的延缓。有效时间范围为：2ms...100h。时间格式见3.4章节。

VII. 对象21-29

这些对象将预设清单上载到产品上，就像您在产品上直接输入和修改一样。但是不可再进一步控制，比如选择一个预设值或在预设值间转换，以便产生电压跳跃。要执行该动作，必须使用其它对象。

3.8.3 关于用户配置文档

PSI 8000和PSI 9000系列产品的特征为，它不只有一个用户配置文件，能存储和读取多达4个配置文件。每个配置文件包含一组设定和一组预设清单。意思是，将预设清单写入产品或阅读他们时，用户必须知道当前选择哪个配置文件。

（如果对象清单是代表多个用户配置文件的产品系列），在[对象清单](#)中必须以颜色表示属于配置文件的设定和数值。.

3.9 产品错误、报警代码和报警类型

错误代码 Error code	缩写 Abbreviation	错误描述 / Error text or description
0		无错误 / No error
1 OV		输出(输入)过压保护 / Overvoltage at output (input)
2 OT		产品内部过温 / Overtemperature inside the device
3 SYS		系统错误 / System error
4 U>		超过电压上限 / Upper voltage threshold exceeded
5 U<		超过电压下限 / Lower voltage threshold exceeded
6 I>		超过电流上限 / Upper current threshold exceeded
7 I<		超过电流下限 / Lower current threshold exceeded
8 SIO2		System Link Mode: 通讯被扰乱 / Communication disturbed
9 MS1		System Link Mode: 一台或多台产品下线 / One or more units are offline
10 S-OV		System Link Mode: 从机报告过压 / Slave is reporting an overvoltage
11 S-OT		System Link Mode: 从机报告过温 / Slave is reporting overtemperature
12 S-PH		System Link Mode: 从机报告电源电压出错 / Slave is reporting mains voltage error
13 S-PD		System Link Mode: 从机减少最大输出功率 / Slave reduces max output power
14 S-?		System Link Mode: 从机不响应 / Slave does not answer
17 F01		内部错误 / Internal error
19 F03		内部错误 / Internal error
20 CAN		CAN: 通讯被扰乱 / Communication disturbed
21 FCT		函数管理器: 不能设置函数 / Function manager: function could not be set
22 UDU		阶跃响应监控: U rise / Step response supervision: U rise
23 UDD		阶跃响应监控: U fall / Step response supervision: U fall
24 IDU		阶跃响应监控: I rise / Step response supervision: I rise
25 IDD		阶跃响应监控: I fall / Step response supervision: I fall
26 PDU		阶跃响应监控: P rise / Step response supervision: P rise
27 PDD		阶跃响应监控: P fall / Step response supervision: P fall
28 PH1		上功率段相位缺少 / Phase loss of upper power stage
29 PH2		下功率段和中功率段相位缺少 / Phase loss of lower resp. middle power stage
30 PH3		下功率段相位缺少 / Phase loss of lower power stage
31 OT1		上功率段出现过温 / Overtemperature of upper power stage
32 OT2		下功率段和中功率段出现过温 / Overtemperature of lower resp. middle power stage
33 OT3		下功率段出现过温 / Overtemperature of lower power stage
34 CC		在UIR操作下: 产品不可在CC模式下操作 / only in UIR operation: device is not allowed to operate in CC mode
35 CP		在UIR操作下: 产品不可在CP模式下操作 / only in UIR operation: device is not allowed to operate in CP mode
36 _		Bat. temp. out of range (电池温度太高)
37 _		Bat. temp. out of range (电池温度太低)
38 _		Bat. voltage out of range (电池电压太高)
39 _		Battery deeply discharged (电池电压太低或已过放)
40 _		Cell fault in battery (电池中单节电池短路)
41 _		Temp sensor fault (温度传感器未连接或出现故障)
42 _		Reverse polarity (电池反接)
43 _		Battery not connected (未连接电池)

 仅在多阶段的产品型号上出现 / only at multi-phase models

 仅适用于BCI 800 R系列 / Applies only to BCI 800 R series

表格: 报警编码

报警管理系统是如何工作的?

系根据左边表格, 将报警类型(如下)区分错误, 并放入报警缓冲区。该缓冲区为对象(见[对象清单](#)分布)。

它适用该规则, 第一个错误会被下一个错误下移至缓冲区。若缓冲区已满, 下一个错误总会覆盖最后的那个错误。读取缓冲区后将自动清空它(针对EL 3000/9000和PS 8000), 或者设定对象54(针对PSI 9000, PSI 8000, PSI 800 R和BCI 800 R)申请下一个动作。

什么是报警类型?

关于**PSI9000**、**PSI8000**、**PSI800R**电源系列和**BCI800R**电池充电器电池充电器的报警、警告和通知的含义与区别, 请参考产品说明指导。其它类型产品, 如**EL3000 / EL9000**电子负载, 仅使用0x01或0x02错误类型。

错误类型:

0x01 - 报警当前被激活

0x02 - 报警不再运行

0x10 - 警告当前被激活

0x20 - 警告不再运行

0x40 - 仅通知

如果产品报警缓冲区被查询并进行分析, 将返回一错误类型, 带一错误代码。报警要优先于警告和通知, 且会覆盖它们, 因此它们被认为不是非常重要或根本不重要的项目。

4. Profibus-现场总线

4.1 基本信息

Profibus是一个在测量和自动化领域有广泛引用的独立现场总线标准。按照IEC 61158标准，它保证不同生产商提供的设备相互间能进行无缺陷通讯。现在市场上的电脑、可编程控制器配有各种各样的接口。

IF-PB1接口支持Profibus DP（即分散型外围周边），特别适合快速现场总线通讯。经RS485连接端进行数据传输，在总线端其运行速度高达12MBaud。

一个Profibus-现场总线可分为多个网段。每一网段由最多32台参与设备组成。总线参与设备可以是主机，从机，转发器和转换器。如果参与设备超过32台，则需用到转发器和转换器。现场总线上的每台设备都要指定一专用地址，于是在第一个和最后一个网段上可连接最多31台参与设备，在中间网段上可连最多30台。而地址0，126和127则保留不用。

结论：在一个现场总线上利用IF-PB1卡可连接共125台从机。

撇开总线线长，可给总线选择9.6kBaud到12MBaud之间的波特率。若根据线长发现总线速度选择错误，则不能保证无缺陷通讯。

IF-PB1接口卡支持所有可用的总线速度，特点是现场总线和产品间的隔离耐压高达1000V，且由产品供电。

4.2 技术规格

订购代码	IF-PB1
通讯配置文档	Profibus DP-V0, Profibus DP-V1
接口类型	9 针 D-SUB 插座
通讯硬件	RS485
网络拓扑结构	线行 (不带转发器), 树型/线型 (带转发器)
参与设备数量	最多31台 (不带转发器), 最多125台 (带转发器)
参与类型	从机
PNO识别码	A000 (hex)

4.3 支持产品系列

IF-PB1支持下列系列产品，也被下面的产品所支持（日期：10/2010）：

- PSI 9000
- PSI 8000
- PS 8000

4.4 主-从通讯

4.4.1 循环

Profibus-DP能辨别主机和从机。主机控制总线通讯，并请求从机给出数据或发送数据给从机。在典型主从系统下，输入数据、输出数据和诊断数据在主机与所有指定的从机之间转换。比如，主机为可编程控制器，会将输入数据存在它内部记忆库内，以后作控制应用。输出数据在下一个传输循环输送给从机。

因此主机总是包含所有从机记忆库内的整套数据，但这些数据要延迟一个循环。

若为IF-PB1卡，实际值与产品状态会恒定地，即循环式地传输给主机。

4.4.2 非循环

与循环式数据转换不同的是，数据也可以非循环地传输。它有一优势在于，可访问现场总线上产品的某单个参数或设定值。非循环数据的转换可将传送数据量减到最低，从而大大减少总线的负载。

4.5 连线/终端

通过一两芯屏蔽线将IF-PB1卡与其它从机连接到总线上。可使用两种不同的符合IEC 61158标准的总线连接线，A型和B型。但不建议用B型线，也不应使用于新产品的安装。

A型线：

- 线材：对绞线，带屏蔽层，1x2
- 波阻Rw：135-165 Ω 在3..20MHz时
- 容量：< 30pF/m
- 环路电阻R：110Ω/km
- 连线横截面：>0.34mm²

传输速度 (kBit/s)	最长线长 (m)
9.6	1200
19.2	1200
45.45	1200
93.75	1200
187.5	1000
500	400
1500	200
3000	100
6000	100
12000	100

用现场总线连线将IF-PB1卡上的D-Sub插座与产品相连。

该插座用于连接输入以及输出总线连线。“输入”和“输出”连接线不能调换，因为只要总线终端在某一机台上运行，则标准的现场总线插头会切断输出连线。断开产品时（断电或与电源断开连接）不会影响总线。这同样适用于配IF-PB1卡的多台产品连接到总线的操作。

激活总线终端电阻可以防止总线末端产生反射波，且给总线提供一个干净、理想的级别。

4.6 传输速度与延时

在4.6.1章节下描述了现场总线接口卡可用的传输速度。关于产品的操作说明，请参考接口卡说明书有关如何访问与更改通讯设定。

传输速度只定义数据在主Profibus机与Profibus从机(即：网关)间以多快的速度传输。Profibus网关芯片在Profibus传输过程中处理数据，并以57600的固定波特率将它们提交给设备。这同样也适用于相反的传输。Profibus上的数据在规定时间内由网关刷新，但是数据本身取决于产品多久刷新一次。这个间隔时间因系列不同而不同。

从机一般会按固定的间隔时间(DP-V0)自动将数据(状态，实际值)输入到总线上，此种情况则为1x每个循环。每次循环时从产品上读取一次的数据会在网关内缓冲一下。一个循环用时约为660ms。

而且，如果该数据是通过DP-V1(设定值等)发送给产品的，也需在每次循环时提交给产品。

但是这会引入一定的**延迟**，直到设定值“到达”产品直流输出端或输入端。这个延迟是由数据传输到网关所花的时间，在网关区的处理时间，传输给产品的时间，产品内的处理时间。例如，想要发送一设定值，同一循环下属于最近设定值的实际值不能读取出来。为什么？因为在第一循环下，设定值是从主机获得然后发送给产品的。在下一个循环下，测量的是实际值。该测得值可能仍不属于新的设定值，因此只有在下一个循环下才可读取。最坏情况是可能要在第三个循环下才能获得对应的实际值。这个与设定如“输出关闭”这样的状态类似，通过DP-V0读取产品状态以便验证输出是否已设为关闭状态。

在的标准配置下，如„set a value and then wait for the corresponding actual value to appear on DP-V0 data“这样的操作通常要花1.2s...1.5s。这个只有采用优化过的脚本才能缩短这个时间，有时可缩短至。这个优化后的脚本在每个循环下仍可传送新的实际值，但是状态和/或报警条件只能每2个或3个循环更新一次。

用户自己可将优化后的脚本上载到网关芯片上，按客户需求可获取这个脚本。

4.6.1 线长对它的影响

用户数据传输速度，Profibus主机与从机间的传输时间，以及线长决定使用无障碍的最大速度。下表列出了IF-PB1接口卡的一般Profibus速度：

传输速度 (kBit/s)	连线最长长度 (m)
9.6	1200
19.2	1200
45.45	1200
93.75	1200
187.5	1000
500	400
1500	200
3000	100
6000	100
12000	100

4.7 操作模式

IF-PB1板上的跳线矩阵用来转换接口卡的不同操作模式。这些模式可以是正常操作，也可以是维护目的。

RS 485/D-Sub端口用于：

- Profibus (现场总线通信)

USB端口(虚拟串行端口)用于：

- 利用WINGATE软件或文本编程工具进行(SPT)文本更新
- 产品控制器的固件更新(更新工具)
- 利用固件下载工具(FDT)进行现场总线栈的固件更新

该接口卡配有一**LED**灯，此灯不亮指示总线连接恰当，通信已准备好。当现场总线上从机控制器初始化序列时，或总线连接出错时才亮。故产品打开后此灯会亮一会儿。

一旦从机从总线上断开，此灯也会亮，以指示出此错误。一般由主机系统的软件配置程序提供现时通信条件的详细描述。

4.8 现场总线地址

每台总线连接设备都要有一个唯一的总线地址。任何产品连到总线前或至少产品在总线上运行前要定义该地址。故建议在安装产品时，按产品说明书的描述，设定现场总线地址(不要与“产品节点”混淆)。任何地址都不可重复。所以建议以升序方式分配地址，地址0保留下来给电脑使用或给产品编程用。

现场总线上所有连接机台地址的分配要与主机系统上项目的地址分配一致。只有这样才能给电源提供一个安全的识别符。

注意：在某特定电源的设置菜单下选定的地址要存储下来，在下次启动产品或按重置按钮重设后可恢复使用。

4.9 对象描述

提示

下面所述在数据报下转换过来的数据有不同类型。设定值和实际值都为整数类型，且含有一百分值。可参考章节1.7和3.3-3.8了解更多详情。

4.9.1 循环式现场总线数据

在此情况下，由从机(IF-PB1)传输到主机上的循环现场总线数据就是实际值和产品状态。但这仅当IF-PB1接受主机传送的配置时才会发生。循环数据转换类型的数据由主机以配置电报形式发送。可从与产品系列相关的对象清单下查出某模块的数据位宽。也可根据某特定模块插槽，在其硬件配置的帮助下获取位宽值。它与可编程控制器的地址宽度是一致的。

对象清单的遵循规则也适用于数据格式和电源的实际值分辨率。给用户应用程序编程时要考虑这些因素。

4.9.2 非循环式现场总线数据

只有当主机发出请求时才发生主从机间非循环数据的转换。IF-PB1卡支持一级与二级主机通信。在章节4.9.2.1下列出的数据报适合这种数据转换类型，并分为多个功能组（也可见产品相关的对象清单）。

4.9.2.1 数据报组

DP-V1产品信息（从机的输出脚）

数据报	槽	指数	长度	类型	清单下的对象*	从主机访问
产品类型	3	0	16	String	0	Read
产品系列号	3	1	16	String	1	Read
产品额定电压	3	2	4	Float	2	Read
产品额定电流	3	3	4	Float	3	Read
产品额定功率	3	13	4	Float	4	Read
产品编号	3	4	16	String	6	Read
产品固件版本	3	5	16	String	9	Read
瞬间设定值	3	9	6	Integer	72	Read
产品报警	3	-	6	Byte	77	Read

DP-V1控制（传送给从机的输入脚）

数据报	槽	指数	长度	类型	清单下的对象*	从主机访问
设置产品条件	3	6	2	Byte	54	Write
设定电压	3	7	2	Integer	50	Write
设定电流	3	8	2	Integer	51	Write
设定功率	3	15	2	Integer	52	Write

报警和实际值（输出脚）

数据报	槽	指数	长度	类型	清单下的对象*	从主机访问
实际设定值 (U,I,P)	2	-	6	Integer	71	Read
装置的状况	2	-	6	Byte	70	Read

* 见另外的[对象清单\(PDF\)](#)，也可见章节3.8。

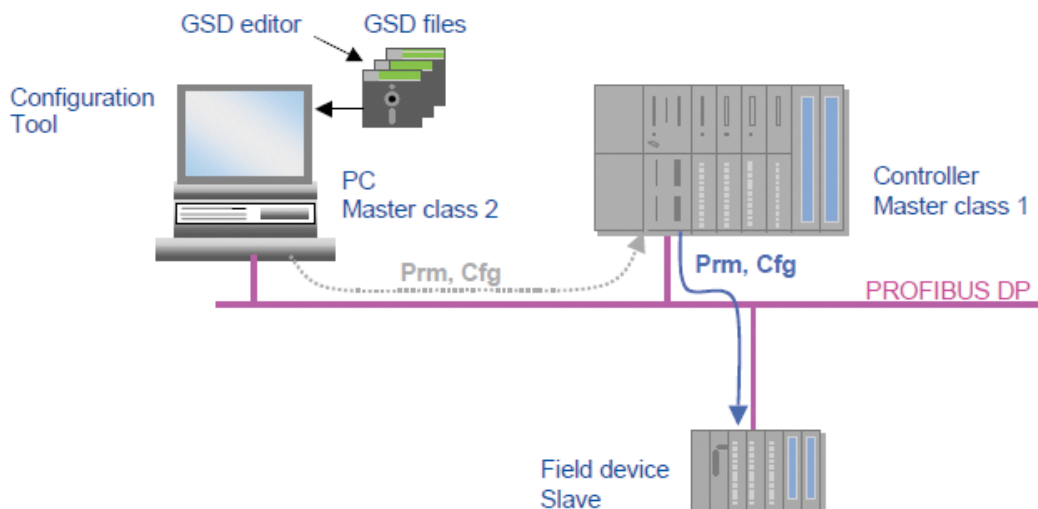


图1: 现场总线工作原理图

4.9.3 数据报的使用

若用户想在其应用中读取或编辑一非循环数据报，意思是用DP-V1主机等级1，需用INDEX（见上面的数据报组，以及SLOT与访问类型（READ或WRITE）选择特定的数据报，。读取时，需知道被读数据的长度，以作为辅助信息。数据长度是PLC软件标准SFBs（系统功能块）的输入参数。

被写数据通常十六进制字节，下面也有范例。如果只有十进制输入，可将它转换成十进制数值。数据的内容在对象清单下定，基于其他接口卡使用的通讯协议。从从机传输过来以及发送给从机的Profibus消息含有数据部分。详情见章节3.2与3.3.2。

4.9.3.1 范例

1) 激活/停用远程控制

如果想要控制产品绝对需要远程控制条件。因此产品状态要更改，并使用。这与对象清单里的对象54有关，字节长度为2。根据对象54，要打开或关闭远程控制，掩码字节必须为0x10，控制字节为0x10。因此0x10 0x10要发送到插槽3，指数6。停用远程控制，就要将0x10 0x10发送给插槽3，指数6。

2) 打开或关闭直流输出

打开或关闭直流输出会改变产品状态，因此要选择„Set device condition“数据报。这与对象54有关，字节长度为2。根据对象54，要打开输出，掩码字节必须为0x01，控制字节为0x01。因此这些字节要发送到插槽3，指数6。将输出设为关闭则需将0x01 0x00发送给插槽3，指数6。

3) 将电压设为40V

这需先将实际电压转化为百分值。假如你有的是一台80V型号，则为50%。如果是一台720V型号，40V就为5.55%。关于设定值的转化详情请见章节„1.7 设定值与实际值的传输“。50%的十六进制值为0x3200，按照0x32 0x00字节将其发送到插槽3，指数7。

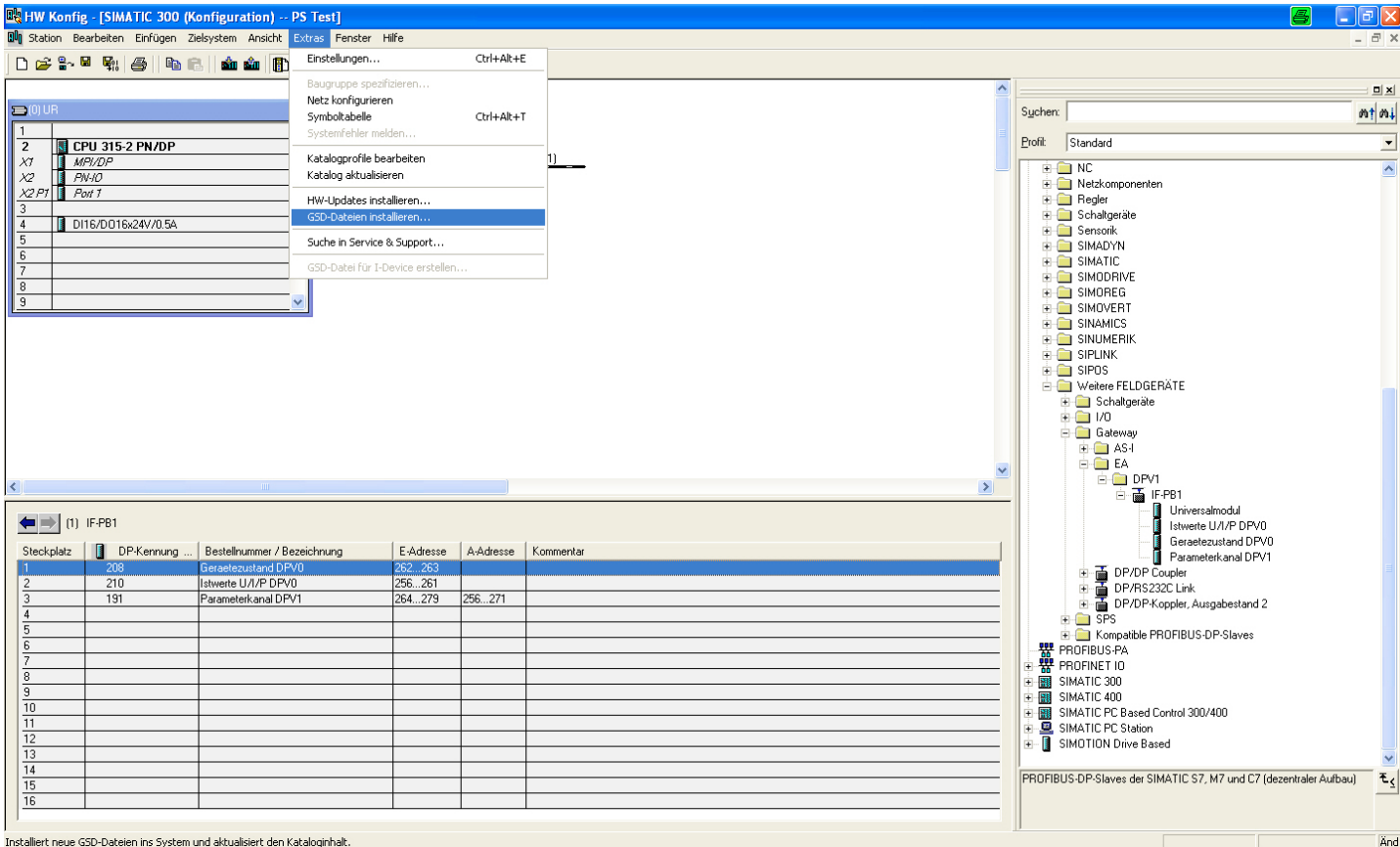


图2: GSE的安装

4.10 以Siemens PLC项目计划为例

4.10.1 将电源整合到现场总线

要通过IF-PB1卡整合一台电源或类似设备，需用到一硬件配置器，这属于Siemens PLC软件包的一个部件。其他公司如Beckhoff或Hilscher，也有配置器。此外，随IF-PB1卡附有一张CD，里面含有一个GSD(Generic Station Device)文档，也可从制造工厂的网站或向制造工厂索取。该文档名称为PB-PSA000.GSD（德文版）或PBPSA000.GSE（英文版）。

该文档用来通报通信模块，要求电源参数发送到用户应用程序的配置器上。于是就定义了总线上的循环数据转换。

4.10.2 GSD文档的安装

刻录于IF-PB1卡附带CD上的GSD文档，通过„Extras -> Install GSD files“菜单指令可将其安装到主机应用下。详情请见下页德文版安装截屏。

文档成功安装后，可在硬件目录下„Profibus-DP -> Other field devices -> Gateway ->EA ->IF-PB1“找到电源的现场总线接口。

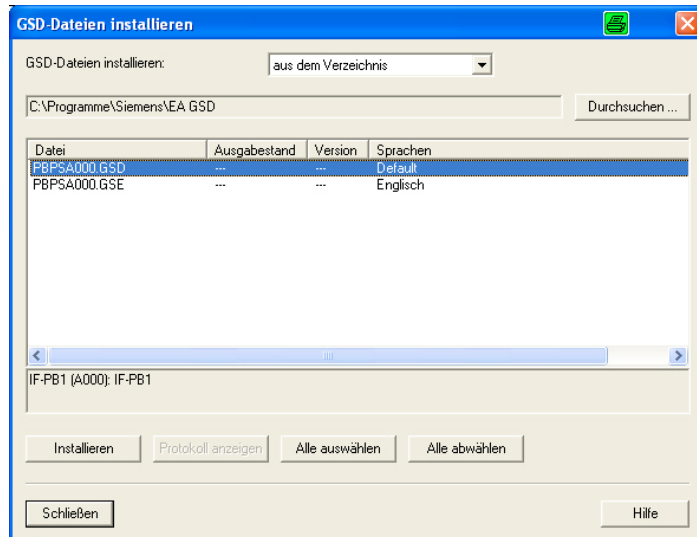


图3: 选择GSE文档安装路径

4.10.3 模块布局

存储于上述（章节）选定路径下的模块要通报给数据转换控制，如拉到或放到模块布局清单下。为使产品可执行全部通讯，模块需按下面规则进行布局：

- 槽1: U/I/P实际值 DPV0
- 槽2: 产品状态 DPV0
- 槽3: 参数通道 DPV1

也可见下图4。

模块布局序列可变化。意思是，当模块必须放于槽3时，“参数通道”与“产品状态”模块可调换。若未放置“参数通道”模块，则不可控制产品。此时将停止非循环数据的转换，且接口面板上的红色LED灯会亮。

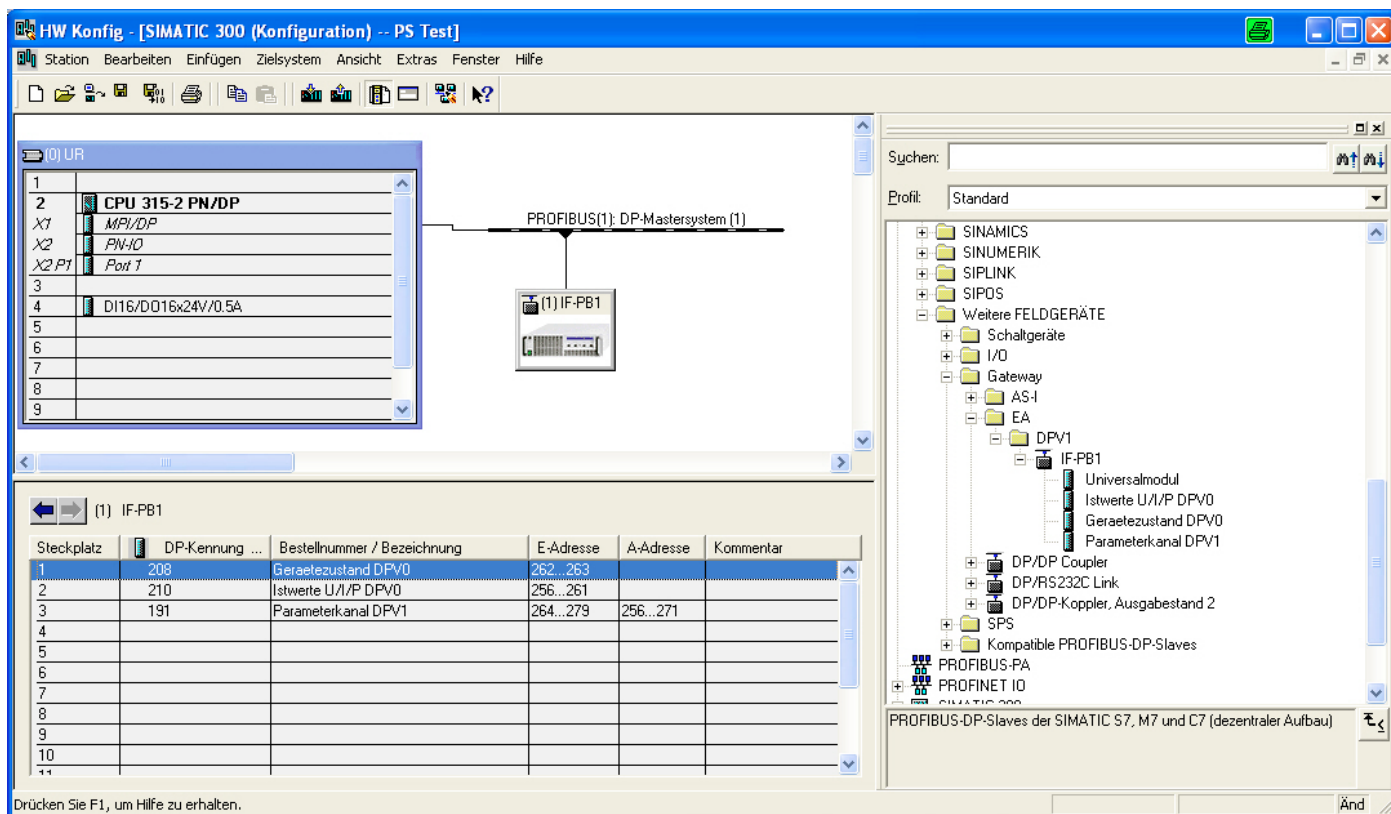


图4: 可能出现的一种模块布局

4.10.4 给循环模块寻址

对于循环数据转换模块一般适用如下：

- 硬件目录下的所有模块只指定一个逻辑地址
- 逻辑地址建立控制周边位置的一部分
- 单模块的周边地址可能会有冲突，故不可分配两次
- 指定的周边地址在用户应用中用于访问数据

4.10.5 给非循环模块寻址

对非循环数据转换模块一般适用如下：

- 周边地址不能与循环模块的任何一地址有冲突
- 因该模块为混合型（输入/输出），输入和输出脚的位址不能有冲突。这是基于西门子通信块（SFB52和SFB53）而设计的，用于非循环数据转换
- 经符合PNO AK 1131的标准接口访问数据时，会先请求一个ID，它由两个位址中较低的那个组成。该位址是非循环数据转换需要的两个信息的基本也是第一部分。
- 第二个信息为将读取或编写的电报号。此处所将的标准为所谓的INDEX

4.11 诊断

4.11.1 标准诊断

标准诊断为6字节电报。该电报在安装于接口与主机间的IF-PB1卡进行初始化时被转换，用户不会察觉到。但是可用工程工具打断该操作。在STEP7执行„Target system -> Show present participants“即完成，执行„Target system-> Diagnosis/Setup -> Component condition“（如下图）选择某特定现场总线设备可显示。如果出现错误，IF-PB1会给现场总线发送另一条消息，此时诊断电报还是永久数据转换的一部分。

见下图5。

针对IF-PB1，诊断电报在正常操作期间运行如下：

0x80 | 0x0C | 0x00 | 0x02 | 0x0A | 0x00

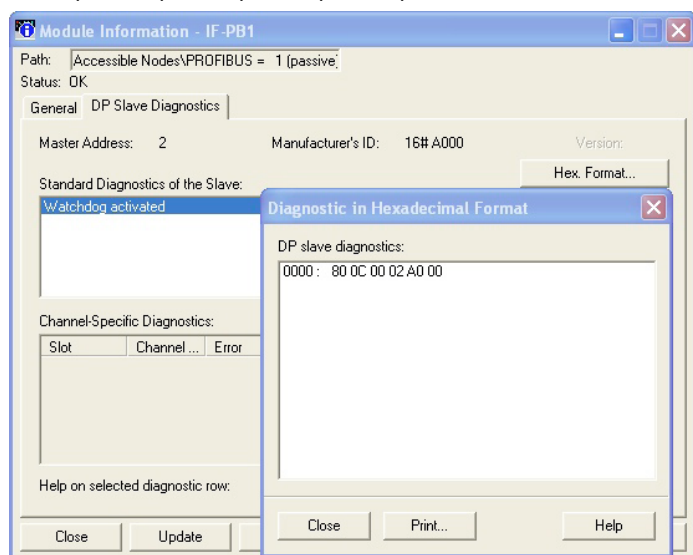


图5：标准诊断

4.11.2 DP-V1 报警管理器（扩展诊断）

按DPV0通信模式描述退出标准诊断时，用户可从IF-PB1接收扩展诊断信息。

在DPV1通信模式下，所谓的基于产品的诊断由报警区和/或状态报告创建而成。激活这个语法，需将参数电报下的DPV1_Enable位元要设为1。

IF-PB1使用状态报告，按下列规则定义：

字节	位元	
1	0	站台不存在（由主机设置）
	1	从机还未准备好进行数据转换
	2	配置数据不一致
	3	从机已扩展诊断数据
	4	从机不支持请求的功能
	5	从机回应无效（由主机设置）
	6	参数化错误
	7	从机已被另外一台不同的主机参数化（由主机设置）
2	0	从机需再次参数化
	1	静态诊断
	2	设为"1"
	3	监视器被激活
	4	收到冻结指令
	5	收到同步指令
	6	保留
	7	从机停止工作（由主机设定）
3	7	诊断溢出 从机上的诊断信息超过电报所需信息
4	0...7	参数化后的主机地址（0xFF 未参数化）
5	0...7	识别码高字节
6	0...7	识别码低字节
7		Header: - Header指示扩展诊断的数据块长度，包括标题字节 - 对于这个模块，该值为0x05（字节7...11 = 5字节）
8		Status_Type: 该值固定为„0x81“，且位元的意义如下： - 位元 7 = 1: „状态“ - 位元 0 = 1: „状态报告“
9		Slot_Number: 该值为„0x00“
10		Specifier - 新出现的错误在Specifier下以„0x00“标注（状态出现） - 消除的错误在Specifier下以„0x02“标注（状态消除） - 若未报告有错误，值为„0x00“
11		用户字节： 下面有详细解释

续下页...

按照上表，IF-PB1的扩展诊断形成如下：

0x05 | 0x81 | 0x00 | 0x00 | 0x??

最后字节以代码形式定义错误信息，或者代表一通信错误（见页面Seite 11页3.7.2章节的表格）或产品错误（见3.9章节下的表格）。

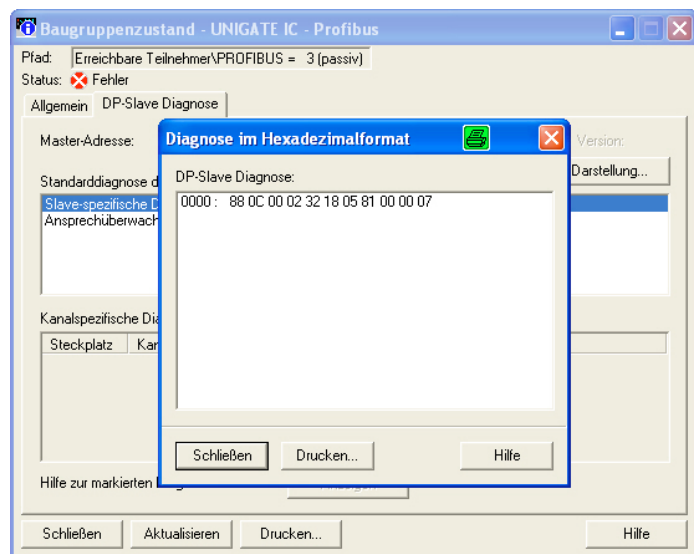


图6 扩展的诊断

	Page
1. General.....	24
1.1 Terms explained	24
1.2 Prologue.....	24
1.3 General notes about the communication	24
1.4 About the USB driver	24
1.5 Structure of the communication	24
1.6 Serial transmission settings	24
1.7 Translating set values & actual values	25
1.8 Telegram structure USB/RS232	25
1.9 Message structure for CAN.....	26
1.9.1 Former CAN system with device node and RID	26
1.9.2 New CAN ID system	26
1.9.3 Split messages.....	26
1.10 Message structure IF-G1	27
1.11 Message structure IF-Ex (Ethernet).....	27
1.12 Message structure IF-Ex (USB)	27
1.13 Timing of messages	27
2. Communication in LabView.....	27
2.1 Overview about the Labview VIs.....	27
2.2 Installation	27
2.3 Short overview about the VIs	27
3. Communication with a device.....	28
3.1 Basic stuff.....	28
3.1.1 Note about the USB driver library	28
3.2 Standard procedure	28
3.3 Building telegrams.....	28
3.3.1 The start delimiter	28
3.3.2 The mask	29
3.3.3 Examples	29
3.4 The time format	29
3.4.1 Time formats for electronic loads	29
3.4.2 Time format for power supplies and battery chargers.....	30
3.5 Tips.....	30
3.6 Trouble-shooting	31
3.7 Error messages.....	31
3.7.1 Explanation about certain error codes	31
3.7.2 Communication error list	32
3.8 Communication object lists	32
3.8.1 Column definition	32
3.8.2 Object examples and explanations	33
3.8.3 About user profiles	33
3.9 Device errors, alarm codes and alarm types.....	34
4. Profibus (IF-PB1).....	35
4.1 General	35
4.2 Technical specifications.....	35
4.3 Supported devices	35
4.4 Master-Slave communication	35
4.4.1 Cyclic.....	35
4.4.2 Acyclic.....	35
4.5 Cabling / Termination.....	35
4.6 Transfer speed and delays.....	36
4.6.1 Influence of cable lengths	36
4.7 Operation modes.....	36
4.8 Profibus address	37

	Page
4.9 Communication with the master	37
4.9.1 Cyclic field bus data	37
4.9.2 Acyclic field bus data.....	37
4.9.3 Using the datagrams.....	38
4.10 Example project planning with a Siemens PLC	39
4.10.1 Integrating the power supply into a Profibus	39
4.10.2 GSD file installation.....	39
4.10.3 Module placement.....	39
4.10.4 Addressing the cyclic modules.....	40
4.10.5 Addressing the acyclic module	40
4.11 Diagnosis	40
4.11.1 Standard diagnosis	40
4.11.2 DP-V1 alarm management (extended diagnosis)	40

**Attention!**

This document does not deal about the text based communication with SCPI commands via GPIB card IF-G1 or network cards IF-E1/IF-E2! There are separate documents for these interface cards.

1. General

1.1 Terms explained

Telegram: Chain of bytes with varying length. Is either sent to or received from the device.

Singlecast: Query or simple message to a single unit. If devices are linked in a chain, like for instance with CAN, this telegram is received by all units, but only accepted by the addressed one. It is related to the start delimiter (1st byte of a message) and device addressing.

Broadcast: Query or simple message to all units. If devices are linked in a chain, like for instance at CAN, this telegram is received and accepted by all units. Can be used to access devices via ports other than CAN by using a device node of 0. For other interfaces than CAN, broadcast type messages can be used to simplify device identification by leaving the device node (2nd byte of a message) always 0, no matter what the device node of the device is set to.

Object: with its properties it describes the object address and initiates defined reactions at the target unit. Comparable with a command.

Message: CAN specific data packet, like a telegram without start delimiter and checksum.

Signal: Part of a message (term used in Vector software)

1.2 Prologue

The communication protocol explained in here with its object orientated telegram structure is very complex. It is thus recommended to use the ready-made LabView components, if the communication is programmed in LabView. The integration into other environments like Visual Basic, C or .NET requires programming knowledge about the setup and use of hardware interfaces like CAN or USB and the addressing of related drivers.

Here we only explain the structure of the data packet (the telegram) and not how it is transmitted correctly.

1.3 General notes about the communication

The firmware of the different types of devices is programmed to consider any circumstances, as far as possible, that may occur when controlling multiple units at once. Thus it is not always possible to perform any action at any time and any state of the device. For example, the data of the function manager of the series PSI 9000 and PSI 8000 (see user manual) are only transmittable in standby state of the unit. Else an error message would be returned, which is pointing the user to the fact that the device is not in standby mode.

1.4 About the USB driver

In Windows XP, Windows 2003, Windows Vista resp. Windows 7 the driver will install two device. One is a USB device called „USB Serial Converter“, the other is a virtual COM port (VCP) called „USB Serial Port“.

The VCP function of the driver is activated by default and will create a new COM port for every USB port that is connected to the PC at least once.

Current LabView VIs (date: 06-2011) can communicate with devices with USB port of either IF-Ux, IF-Ex or IF-PB1 cards only via the virtual COM port. Apart from that, the VIs support the Ethernet port of IF-Ex cards and GPIB (IF-G1 card).

More recent versions of the driver, than the one on the included CD, can be found on the web site of the USB chip manufacturer FTDI at www.ftdichip.com.

1.5 Structure of the communication

The communication with the controlled units is based on these telegram types:

a) Message: an object is sent which shall, for instance, set the output voltage. As long as this action is permitted by the current state of the device, the object is accepted and executed. The device won't send any reply. If it's not permitted it will send a reply in form of an error message.

b) Query: a query is sent by using a certain object, for instance „get actual values“, and a reply is expected. If the query is permitted for the current state of the device it is executed and replied. The reply contains the requested data. If not permitted it will send an error message as reply.

1.6 Serial transmission settings

Subject: RS232 and USB ports when using VCP (also see section 1.4).

With the serial transmission of one byte following bits are sent:

Start bit + 8 Data bits + Parity bit + Stop bit

The parity is checked for „odd“.

The USB port of IF-Ux, IF-Ex and IF-PB1 is internally working with RS232 characteristics. For these card types it is required to set at least these communication parameters for the particular driver and, when using a RS232 card, on the device, too:

Baud rate RS232 card:	9600Bd ... 57600Bd
Baud rate USB card :	56700Bd
Parity:	odd
Stop bits:	1

1.7 Translating set values & actual values

The set values and actual values (see external [object lists](#)) are, with a few exceptions, transmitted as percentage values, whereas 0x6400 corresponds to 100.00%. Hence any real set value has to be translated before and any received actual value after transmission.

If a device has a nominal voltage of 80V and the queried actual value is 0x3200 (0x3200 = 50.00%) then it corresponds to 40V output voltage.

The high byte is the percentage number (0x64 = decimal 100) and the low byte is the decimal places of it.

$$\text{Real value} = \frac{\text{Nom. value} * \text{Per cent value}}{25600}$$

Example: Max. value of the device is 80V, the percentage actual value came in as 0x2454 = 9300. It results in: Actual value = (80 * 9300) / 25600 = 29.06V

$$\text{Value to send} = \frac{25600 * \text{Real value}}{\text{Nom. value}}$$

Example: the set value for power shall be 500W, the max. value of the device is 640W. With the formula it results in:
Percentage set value = (25600 * 500) / 640 = 20000 = 0x4E20.

1.8 Telegram structure USB/RS232

The interface cards IF-Rx (RS232) and the USB ports of IF-Ux, IF-Ex and IF-PB1 are using a slightly different telegram structure than the CAN cards IF-Cx. Skip to section 1.9 if you're using a CAN card.

The telegram is structured like this

SD + DN + OBJ + DATA + CS

and is built of these bytes:

Byte 0: SD (start delimiter)

The start delimiter determines how to handle the telegram furthermore. Meaning of the bits:

Bits 3-0: Data length (Bytes 3-18)

These define the data length - 1 of the data in the telegram. At a query, the data length of the expected data is given here.

Example: the [object lists](#) states a data length of 6 for a certain object, so the lower nibble of SD will be 5.

Exception: object of type „string“ do not care for the data length as long as it is not higher than given in the object list, so any value between 0 and object length can be given here.

Bit 4

- 0= Telegram from device to PC
- 1= Telegram from PC to device

Bit 5

- 0= Singlecast, telegram to a certain device node
- 1= Broadcast telegram to all device nodes (with CAN), else like singlecast with DN = 0)

Bits 7+6: Transmission type

- 00 = Reserved
- 01 = Query data
- 10 = Answer to a query
- 11 = Send data

Byte 1: DN (device node)

The device node identifies and addresses devices inside a bus system like CAN or GPIB. Each node number must only be assigned once. This is used to address a particular device. Value range: 1...30, others are invalid. Using CAN, the CAN IDs are calculated from the device node (depends on the device type and firmware, too). See section 1.9 for details. In point-to-point systems like RS232 or USB, this address is only important when using message type „single cast“ (see Byte 0). When using broadcast, the DN can always be 0.

Byte 2: OBJ

The communication objects for a device are addressed by this byte. In the communication [object lists](#), the objects and their function(s) are explained in detail.

Byte 3 - 18: Data field

The data field can be up to 16 bytes long, hence the length of the telegram varies. If a query is sent (PC → device) there will be no data (=data field empty) and the checksum of the telegram (see below) follows directly after byte 2. Only if an answer (device → PC) is sent, an expected one or an error, there will be data of a specific length.

Last 2 bytes: CS (checksum)

The checksum is always located at the end of the telegram. It is built by the simple addition of all preceding bytes of the telegram and attached to the end. It is two bytes long (16 bit value). The high byte is placed before the low byte.

Example of a telegram:

Object no. 71 (query actual values) shall be sent to a device with device node 1. The telegram has to be like this in hexadecimal values:

55 01 47 00 9D

And the answer could look like this:

85 01 47 64 00 1E 00 50 00 01 9F

(this translates to 80V, 30A and 2400W actual values for a 80V, 100A and 3000W power supply, like the PSI 9080-100)

Also see section 1.7 for the conversion of set values and actual values.

For more examples see sections 3.4 and 3.8.2.

1.9 Message structure for CAN



Attention!

The series PS 8000 (from firmware 6.01), PSI 8000 (from firmware 3.14) and EL 3000/EL 9000 (from firmware 5.01) feature two CAN ID systems, as described below. Other series only support the system as described in 1.9.1.

The interface cards IF-Cx support the CAN V2.0a standard. The extended address format is not used.

Refer to the external user guides of your device and the interface card for the CAN specific communication settings.

1.9.1 Former CAN system with device node and RID

Note: this CAN system is implemented into every device series that supports a CAN interface card type IF-Cx, except in PS 8000 models with firmware 6.01 or up.

The CAN driver chip requires the **identifier**, up to 8 data bytes and the **data length** for a transmission. The **identifier** is 11 bits long (CAN 2.0a) and specified by the **device node**, the relocatable identifier segment (**RID**) and the type of the message. For every unit there are two identifiers:

[RID*64 + device node * 2] (1st identifier) and

[RID*64 + device node * 2 + 1] (2nd identifier),

whereas the 1st identifier is used to send messages to the device to set something and the 2nd one to send messages in order to query something and to receive replies.

A message can contain a maximum of 8 bytes. The first byte is the number of the communication object. After this you can put or receive up to 7 data bytes (see communication [object lists](#)).

In order to send an object with a 16 bytes long data field it is thus required to send at least three message and the data field has to be split up over those three messages. See section 1.9.3 for more.

Two examples:

a) The device shall be set to remote control mode. This is required to control the device by a status command or to set values. The **device node** was set to 15 and the **RID** to 3. The message is of „send only“ type. The send identifier calculates as $3 * 64 + 15 * 2 = 222_{10}$ or $0xDE$ with the above formula. According to the [object lists](#) we use object 54 (hex: $0x36$) with the data bytes $0x10$ (mask) and $0x10$ (set remote). The resulting data length is 3, so the CAN data results like this:

```
ID  DL  DATA
DE  03  36 10 10
```

b) In case you don't want to set the state but query it, the identifier $0xDF$ is used (2nd identifier) and because it is a query, the object number is sufficient as data. The CAN data are thus like this:

```
DF 01 36
```

and the answer should be like this:

```
DF 01 36 10 10
```

1.9.2 New CAN ID system

Note: the new CAN ID system is available as firmware update for series PS 8000 since June, 2011. Other series will follow. With a firmware update which contains the new CAN ID system, the former CAN ID system is not supported anymore and custom PC software probably stops working.

The new CAN ID system uses 3 basic IDs plus a broadcast ID (per unit) for communication. It is Vector software compatible, such as CANalyzer or CANoe. Vector databases in format *.dbc are available upon request or on the CD that is included with the CAN interface card. There will also be ready-to-use configurations to be opened in those softwares for demonstration purposes.

In the device setup and if CAN interface IF-C1 or IF-C2 is equipped, the user adjusts a base ID. Then the device uses three CAN IDs (base ID, base ID + 1 and base ID + 2). Those IDs have to be unique in the bus and thus the base ID adjustment is done in steps of 4. If multiple units of the same type are used within a bus, they all need to have a different base IDs.

Regarding the broadcast ID, this is vice versa. Here it is intended for multiple units to have the same broadcast ID adjusted. The purpose is to send the same message to x units on the bus, in order to set, for example, the same current set value at the same time. The broadcast ID is adjusted separately from the base ID in steps of 1 and has to be unique, too. It means, it has to be different from the other CAN IDs of the device.

The three standard CAN IDs are assigned as:

Base ID = Used by the PC to only send messages that set a value or condition and don't create an answer telegram

Base ID + 1 = Used by the PC to query anything from the device by just sending the queried object (see documentation about the available objects (i.e. commands))

Base ID + 2 = Used by the remote device to send answers to queries or to send error messages

The broadcast ID is assigned as:

Broad ID = Used by the PC to only send messages to multiple units, in order to set a value or condition and don't create an answer telegram

1.9.3 Split messages

A split message is a message, which is split into multiple submessages (only concerns objects in „string“ format). An extra identifier is inserted here, after the object number (=object address). The extra identifier of the first message is $0xFF$, for the second message it is $0xFE$ and $0xFD$ for the third one. The order of these messages is not specified. The telegram has to be composed again later from these messages. When using the gateway function the split telegrams are not composed by the gateway. This has to be done by a superior control unit.

1.10 Message structure IF-G1

The message structure for the text based communication via a GPIB card is described in external user guides.

Link: [SCPI command list for power supplies](#)

Link: [SCPI command list for electronic loads](#)

1.11 Message structure IF-Ex (Ethernet)

The message structure for the text based communication via the Ethernet port of an IF-Ex is described in external user guides.

Link: [SCPI command list for power supplies](#)

Link: [SCPI command list for electronic loads](#)

1.12 Message structure IF-Ex (USB)

See sections 1.6 and 1.8. Communication in LabView

1.13 Timing of messages

After every query the device typically needs between 5ms and maximum 50ms for the answer. Basically you are allowed to send queries directly after another. But if an event was received it is required to wait at least 50ms. A time of 100ms is recommended in order to not slow down the device's operation by too much communication.

When using the gateway function (only supported by PSI 9000 series) you need to consider the time that will be consumed by transferring the telegram from one bus system to the other. The answer may be delayed up to 200ms here.

After receiving an error message over this gateway you should consider to wait at least 100ms until the next transmission.

Broadcast with CAN

After every broadcast query all bus sharing units can only answer consecutively. Depending on the bus system, the baud rate and the number of units, as well as the extra bus traffic, the answers can be delayed more or less. The time is not specifiable and can only be estimated by the formula *bus sharing units * response time at singlecast*. In most cases the response time will be shorter.

2. Communication in LabView

2.1 Overview about the Labview VIs

For an easy integration of multiple and even different devices into existing LabView applications we provide a set of Labview VIs.

Those virtual instruments (VI) enable a simple implementation into and the programming of an application without the need for the user to learn about the lower levels of communication.

In order to use the functionality of these VIs it is required to use and run the software development tool LabView from the company National Instruments. The LabView VIs require version 7.0 or higher.

Following minimum system requirements have to be considered:

- Pentium 3 CPU with 256 MB memory
- Windows operating system (Win98 and WinXP)

Updates of these VIs can be downloaded from our website, if available, or are sent via e-mail upon request.

2.2 Installation

To install and use the VIs in LabView in your environment, please read the file „installation_english.pdf“ on the included CD for instructions.

After the installation you can find the VIs in the context menu of the LabView IDE in „Instrument I/O -> Instrument drivers -> IF-XX“.

Some VIs are only for a specific series and will only work with these. They are located in subfolder of the installation. The other VIs are for common use. Usage and functionality are described in the user guide of the VIs. You can access it the usual way via the LabView context help (Ctrl+H) or open the CHM directly. Depending on the Windows version it can be necessary to copy the help file to a local hard drive in order to read it correctly.

Note

Always use the proper VIs for your device and for your LabView version.

Please read the LabView VIs help file on the included CD in order to get an overview and a clue about the handling.

2.3 Short overview about the VIs

Date: 08/2012

After unpacking the VI set, several subfolders become available.

The common VIs in folder „\Common“:

- **Device_close.vi** - closes the communication with a specific device. Before exiting your application, any open communication port should be closed, if device communication is not used any further.
- **Device_scan.vi** - scans certain hardware ports (RS232, GPIB, Ethernet, USB(VCP)) for compatible devices and returns the number of found devices as well as a list of device information, which is required for the other VIs.

- **Device_select.vi** - is used to select and open a specific device from the list of devices that is returned by „Device_scan.vi“. Multiple devices can be selected and used in parallel.

Series specific VIs in the folders:

- „\BCI8 Series\ - for models of series BCI 800 R
- „\PS2 Series\ - for models of series PS 2000 B
- „\PS8 Series“ - for models of series PS 8000
- „\PSI8 Series“ - for models of series PSI 8000 and PSI 800 R
- „\PSI9 Series“ - for models of series PSI 9000
- „\EL Series“ - for models of series EL 3000 and EL 9000

The examples in folder „Examples“ are intended to demonstrate the use and placement sequence of the common and specific VIs. The series specific VIs in the example VI can be replaced by other specific VIs in order to test a different device type.



Attention!

Whenever a series specific VI is replaced by another series specific VI, the enum constant on input „command list“ must be deleted and created again!

3. Communication with a device

3.1 Basic stuff

The following sections deal about the composition of the communication telegrams, the dependency of the communication from the state of a device and the problems related to that topic, without explaining how to use the USB driver when using the USB port in low level or how to correctly send a CAN message. This has to be learned and done by the user.

3.1.1 Note about the USB driver library

When using the USB card IF-Ux or the USB port of the Ethernet cards IF-Ex or Profibus card IF-PB1, a USB driver has to be installed. After this, the user can choose to communicate via the USB driver on low-level access or via a virtual COM port (VCP), whereas accessing the VCP is much simpler.

On the included CD in the folder \manuals\other\ftdi, there is a PDF which describes the functions of the USB driver DLL in detail. In general it applies, that a device (in this case the USB hardware) has to be opened first (FT_Open or similar), then configured (FT_SetBaudRate, FT_SetDataCharacteristics etc.) and then it can be written (FT_Write) or read (FT_GetQueueStatus, FT_Read). As soon as the device is not used anymore it is advised to close it (FT_Close), while it is advised not to open and close it for every read-write cycle. Configuration of the USB hardware needs to be done only once as long as it is powered. The functions FT_Write and FT_Read serve to transport the actual telegram bytes of the object orientated communication described in the following sections.

3.2 Standard procedure

The programming of the various devices, in which the interface cards are used, always follows the same scheme. It only differs in number and functionality of the communication objects that are supported by a specific device series.

Generally applies:

- Monitoring, i.e. only querying actual values and status, is always possible. The devices don't require the remote mode.
- Setting of status and set values is considered as controlling and requires the activation of the remote mode (remote in this case means that the device is remotely controlled via a digital interface card)
- The remote mode can be blocked by certain circumstances. For instance, the explicit local operation (only PSI 9000) or a different mode the device is in and which does not allow remote control. For further details refer to the user manual of your device.

In order to start controlling a device, for example by sending a set value, you need to

1. activate the remote mode (object 54)

2. send the set value

and, if not already done,

3. set the output/input to on

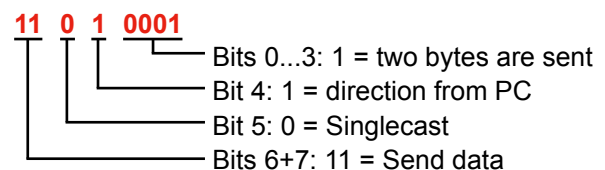
The remote mode should be left again, if not used any further. As long as it is active, the device can not be operated manually or only restrictedly. The mode is indicated on the front.

3.3 Building telegrams

3.3.1 The start delimiter

Note: the start delimiter is not used with CAN

According to the telegram format (also see 1.8), the first byte is the start delimiter (SD), which depends on the type and direction of the telegram. For this example the SD will be 0xD1, and look like this in single bits :



Alternatively to the bitwise assembly, this can be simplified by using hex values. Starting from bits 6 + 7 we get:

SD = Message type + Cast type + Direction + Length

whereas the message type is either

0xC0 Send data or
0x40 Query data

and the cast type is either

0x00 Singlecast or
0x20 Broadcast

and the direction is either

0x10 from PC to the device or
0x00 from device to the PC

and the data length - 1 can be

0x00...0x0F up to 16 bytes of data (at CAN see „1.9.3 Split messages“)

Note

Always note, that the data length is defined as number of bytes to send -1!!!

3.3.2 The mask

Some objects require a mask, which is sent before the control byte. The possible mask value is given in column 7 of the [object lists](#). The bits of the control byte have various functions, so it has to be determined which bit(s) have to be changed. This is determined by the mask with a 1 for every bit that is shall be changed in the control byte. Example: bit 4 shall be changed to 1. Then the control byte would be 0x10 and the mask 0x10, according to the value of the bits. In case there is only one mask given, like 0x0F, it has to be always used as 0x0F. Then all concerned bits are changed at once, for example like with object 56 (function manager of PSI 8000 and PSI 9000).

Attention!

In order to avoid logical collisions of functions that are assigned to the bits, it is advised to only set 1 bit per telegram!

3.3.3 Examples

Example 1: Switch device to remote mode

The address (node) of the contacted device is set to 5, the object to use is 54 (hexadecimal: 0x36), the mask for the remote mode (also see [object lists](#)) is 0x10 and the control byte has to be 0x10 (remote = on). Then we get this telegram:

D1 05 36 10 10 01 2C

In order to reverse this command, i.e. deactivation of the remote mode, you need to send **D1 05 36 10 00 01 1C**. The mask stays the same, only the control byte changes.

Example 2: Querying actual values via CAN card

Using CAN, the start delimiter SD and the user-calculated check sum CS are not used. So we only need the object (according to the [object lists](#) it is 71 (hex: 0x47)), the identifier ID and the length of the bytes to send. In a CAN message, the object is included in the data length, so this message would have a data length of 1, because we only send the object that queries the actual values. Using the old CAN system (see sections 1.9.1 and 1.9.2), the device address (node) is 5, the RID is 8. According to the formula from section 1.7 the identifier calculates as $8 * 64 + 5 * 2 + 1 = 523$ (hex = 0x20B). The +1 is because it is a message of type „query“.

We now send one byte to ID 0x20B. The CAN message could look like this (depending on how it displayed on the Windows GUI that is used):

02 0B 01 47

Object 71 (0x47), queries actual values
Data length = 1
Identifier



Attention!

This is NOT the bit combination of the CAN message that is actually sent over the CAN bus. A CAN controller merges various bits into it and adds an own checksum to it. These are only the bytes that are submitted to the CAN driver.

An answer to this query could look like this:

02 0B 06 64 00 0A 00 42 AA

Same identifier, data length is 6, because three actual value of 16 bits size each are sent. The actual values are transmitted as percentage values and need to be translated to real values. See section „1.7 Translating set values & actual values“ for details. For an EL 9080-200, the actual values would translate to 100% for voltage (=80V), 10% for current (=20A) and 66,7% for power (=1600W).



Note

The nominal values for power, current and voltage can be read out from the device with the proper objects and used to translate the actual values to real values.

3.4 The time format

The time format represents times from 1µs to 100h by a 16 bit value. Such time stamps are checked by the device for being correct. Values that are too high or too low are not accepted and will return an error message. The upper 4 bits are used as a mask to determine the time range, the rest of the bits represent the time value.

The time format is used to write (i.e. set) or read time values.

It applies to any device that features a function related to time, as long as this time value is settable. The resolution of the time ranges in the table below does not necessarily match the resolution of the particular device. In this case, the values are rounded down. An example: a time value of 0x23E7 is sent. This represents $999 \times 1\mu s = 999\mu s$. The manually adjustable time value of the device in this time range is but 0.95ms or 1ms. The 999µs are rounded down to 950µs. Hence there will be 0x23B6 returned (=950) when read back, instead of the sent 0x23E7.



Note

Not all devices use all of the masks in the table below.

3.4.1 Time formats for electronic loads

For electronic loads and the **rise time (object 92)** applies, according to the main table below:

Time range		Step width of device	Mask	Value range**	
from	to			from	to
30µs	99µs	1µs	0x2000	0x201E	0x2063
0.10ms	0.99ms	10µs	0x2000	0x2064	0x23DE
1.0ms	9.9ms	100µs	0x3000	0x3064	0x33DE
10ms	99ms	1ms	0x6000	0x6064	0x63DE
100ms	200ms	1ms	0x7000	0x7064	0x70C8

** Values differing from the step width are rounded

For electronic loads and the **pulse width (objects 90 and 91)** applies, according to the big table below:

Time range		Step width of device	Mask	Value range**	
from	to			from	to
0.05ms	0.95ms	50µs	0x2000	0x2032	0x23B6
1.00ms	9.95ms	50µs	0x3000	0x3064	0x33E3
10ms	99.9ms	100µs	0x6000	0x6064	0x63E7
100ms	999ms	1ms	0x7000	0x7064	0x73E7
1.00s	9.99s	10ms	0x4000	0x4064	0x43E7
10.0s	100s	100ms	0x9000	0x9064	0x93E8

** Values differing from the step width are rounded

For electronic loads and the **elapsed time of battery test (object 64)** applies, according to the big table above:

Time range		Step width of device	Mask	Value range**	
from	to			from	to
1s	3599s	1s	0x8000	0x8001	0x8E0F
1h	99h:59m	1m*	0xC000	0xC03C	0xD76F

* Above 1h, time can only read as HH:MM

** Values differing from the step width are rounded

Example 1: the rise time for an electronic load shall be set to 75ms. The step width of load's the time range where the 75ms belong to, is 1ms. So we need to use the 0x6000 time range. Time range resolution here is 0.1ms, so it results in a time value of 750 (75ms : 0.1ms). This translates to 0x2EE. Together with the mask you get a value of 0x62EE as time value for the rise time (object 92).

Note

LabView users need to provide the time in a different way. See VI documentation.

Example 2: the time value of the battery test has been read and shall now be translated to the normal time format. The overall resolution of the battery test time is 1s. Since the time ranges allow 1s resolution only up to 1h, the time above 1h is given in minutes and hours. A value of, for example, 0x8743 would translate into 1859s or 30m59s, whereas a value of 0xC532 would translate to 1330m or 22h10m. The seconds are omitted in this time range, so you would always read out the same time value during 1 minute.

Example 3: Setting the pulse width for A (object 90) to 5s. According to the 2nd table from above the time range mask is 0x4000. In combination with the resolution of 10ms for this time range, a value of 500 (5s : 0.01s) results, in hex 0x1F4. The total resulting time value then would be 0x41F4.

Mask *	Time value (bits 11..0)				Resolution	Resulting time range
	min.(dec)	min.(hex)	max.(dec)	max.(hex)		
0x2000 ⁽¹⁾	0	0x00	999	0x3E7	1us	0 ... 0,999ms
0x3000 ⁽²⁾	100	0x64	999	0x3E7	10us	1ms ... 9,99ms
0x6000 ⁽¹⁾	100	0x64	999	0x3E7	100us	10ms ... 99,9ms
0x7000 ⁽²⁾	100	0x64	999	0x3E7	1ms	100ms ... 999ms
0x0000 ⁽¹⁾	0	0x00	4999	0x1387	2ms	0 ... 9,998s
0x4000 ⁽¹⁾	100	0x64	5999	0x176F	10ms	1,00s ... 59,99s
0x8000 ⁽¹⁾	1	0x01	3599	0xE0F	1s	1s ... 59min:59s
0x9000 ⁽²⁾	100	0x64	1000	0x3E8	100ms	10,0s ... 100,0s
0xC000 ⁽¹⁾	0	0x00	5999	0x176F	1m	01:00h ... 99:59h

Table: Time format

* If the mask is used to translate time values into real time, either bits 15...13 or 15..12 are relevant, depending on the used time range

3.4.2 Time format for power supplies and battery chargers

Applies to: PSI 8000 / PSI 9000 / BCI 800 R

Time values for sequences of the PSI 8000 / PSI 9000 function manager are time formats in 2ms grid or a multiple of it. Other time values also use these time formats, like with objects 39-43 and 47.

Time range		Step width of device	Mask	Value range	
from	to			from	to
0.002s	9.998s	2ms	0x0000	0x0001	0x1387
10.00s	59.99s	10ms	0x4000	0x43E8	0x576F
1:00m	59:59m	1s	0x8000	0x803C	0x8E0F
1:00h	99:59h	1m	0xC000	0xC03C	0xD733

Note

Not all time values that can be read from a device are necessarily build by the time format specification. See [object lists](#).

3.5 Tips

I. Detecting a device node (not with GPIB)

If you want to, for example, control a device via USB and you don't know the device's node, you could for example use the broadcast node 0 and query the device class. The device or the devices will answer with its/their own device node, that has/have been set at the device(s). The device node(s) can be furthermore used to control and distinguish the devices.

Using an USB or RS232 interface is a point-to-point connection and here you can generally use broadcast message type with device node 0 (see section „3.3.1 The start delimiter“).

II. Remote and standby

The object 54 is used to either activate/deactivate the remote control or switch the input/output of a device. The object can be used to activate both at once, but it is strongly not recommended to do so, because setting the input/output requires the remote control operation already being active and else would generate an error message. You should rather activate remote control first with the corresponding bit set in the control byte and then control the input/output by sending object 54 a second time with a different control byte. When deactivating remote control it simply goes vice versa.

It is also useful to read back the state of the device with object 70, in order to check if object 54 has been set correctly.

3.6 Trouble-shooting

Problem: The device does not react or respond to commands

Possible causes with USB

- The USB card requires a driver. Check if the driver is installed correctly and if you can find the card as „USB Serial Converter“ in the Windows device manager in the section of „USB controllers“.
- The wrong device node (=address) is used to communicate with the device.

Possible causes with RS232

- You are not using a 1:1 cable for the RS232 card.
- The wrong device node (=address) is used to communicate with the device.
- Device and PC are configured for different baudrates etc.
- The communication cable is too long for the configured baudrate (also see section „2. Technical specifications“ of your interface card user guide).

Possible causes with GPIB

- If multiple device are connected to a IEEE bus, one or more device addresses might be double.
- A wrong syntax is used. For example, an electronic load does not react to the command OUP, because it features an input. Or the command was not valid for the type of device you tried to contact.

Possible causes with CAN:

- The wrong CAN ID is used.
- Wrong baudrate set
- Wrong sample point selected (only at PSI 9000, see user guide)
- The device is located at the end of the bus and is not terminated

Problem: Multiple queries were sent, but not all of them were answered

Cause: The queries have been sent subsequently too fast. Depending on the communication type and speed and the execution time of the device, you need to include a certain latency between two queries.

Rule of thumb: Latency = Transmission time + Execution time

The execution time lies at typ. 5-20ms, depending if there only was a query or if something has to be set. The transmission time can be calculated from the baudrate and number of bits that are sent.

Problem: Set values and status are not set

Possible causes

- The contacted device is not in remote control mode or can't currently be set to this mode, because it might not be allowed in this very moment or any other condition for setting the device into remote control is not fulfilled
- The sent values are wrong (too high, too low) or the standard value range (0...0x6400 for voltage, current etc.) is additionally limited by limit values (only with PSI 9000 / PSI 8000). An error message is sent in this case.

3.7 Error messages

The table in section 3.7.2 lists all possible error codes that could be returned by the device with an error message. The code is used to point the user to the origin of the error. Some errors are caused by erroneous messages (i.e. wrong checksum etc.), others might come from the device, denying a command in dependency of the current condition.

Error messages are in telegram format, i.e. they are composed of a start delimiter (except with CAN), object number (to identify an error, **0xFF** is used as object number), error code in data field and checksum (except with CAN).

Example: in case you want to set the voltage with object 50 and the device is not in remote control, you would receive the error message **C0 07 FF 09 01 CF** from a device with device node 7.

3.7.1 Explanation about certain error codes

Code 0x7: the object number used in the telegram is unknown to the device. This is because not all device types use the same object numbers. Use the object list for you device to compare.

Code 0x8: the length of the data field in the telegram is defined in the [object lists](#). This error code will be returned if a set value, which is always 2 bytes because of type „int“, should have been sent but the data field only contained one byte. Even if the start delimiter contained the correct telegram length. This is a protection against setting wrong values.

Code 0x9: an object has been sent in order to set a value, but the device is not in remote control mode. In this condition you only have read permission, but no write permission. You need to set the device to remote control mode first.

Code 0xE: Strings have to be transferred differently when using CAN. If the string length is greater than 8 characters, you have to use split messages that are designated with the string start tokens 0xFF, 0xFE etc. Also see „1.9.3 Split messages“.

Codes 0x30/0x31: these are related to set values. All set values have an upper and a lower limit, which are defineable at the power supplies. The default upper limit for e.g. the current set values is 0x6400 (=100%) and the lower limit is 0. Limits also apply to time values.

Code 0x32: a time value using the wrong time range has been sent. The upper or lower limits are not exceeded by the value, but it still causes this error.

Codes 0x36: Conditions for the access to these data are not fulfilled. See [object lists](#) about the access conditions in its column 4.

3.7.2 Communication error list

Error code		
Hex.	Dec.	Description
01	1	RS232: Parity error
02	2	RS232: Frame Error (Startbit or Stopbit incorrect)
03	3	Check sum incorrect
04	4	Start delimiter incorrect
05	5	CAN: max. nodes exceeded
06	6	Device node wrong / no gateway present
07	7	Object not defined
08	8	Object length incorrect
09	9	Read/Write permissions violated, no access
0A	10	Time between two bytes too long / Number of bytes in message wrong
0C	12	CAN: Split message aborted
0F	15	Device is in "local" mode or analogue remote control
10	16	CAN driver chip: Stuffing error
11	17	CAN driver chip: CRC sum error
12	18	CAN driver chip: Form error
13	19	CAN: expected data length incorrect
14	20	CAN driver chip: Buffer full
20	32	Gateway: CAN Stuffing error
21	33	Gateway: CAN CRC check error
22	34	Gateway: CAN form error
30	48	Upper limit of object exceeded
31	49	Lower limit of object exceeded
32	50	Time definition not observed
33	51	Access to menu parameter only in standby
36	54	Access to function manager / function data denied
38	56	Access to object not possible

Legend




	Communication error
	User error
	Internal error

Table: communication errors

3.8 Communication object lists

Important! The object list for your device is located in an external PDF file and should be distributed along with this document.

Link: [Object list series PSI 8000 T / DT / 2U / 3U](#)

Link: [Object list series PSI 9000](#)

Link: [Object list series PSI 800 R](#)

Link: [Object list series BCI 800 R](#)

Link: [Object list series EL 3000 und EL 9000](#)

Link: [Object list series PS 8000 T / DT / 2U / 3U](#)

The object lists are a reference for programming custom applications which shall control our device remotely and apart from LabView, where ready-to-use LabView VIs are available. LabView users wanting to reconstruct the binary telegram communication also use these lists as reference.

3.8.1 Column definition

The **1st column** contains the object number (=command) as a decimal value. This number has to be assigned to the byte **OBJ** in the telegram.

The **3rd column** defines if the object is read only (ro), i.e. it can only be queried from the device or if it can also be written (w, r/w). Reading is always possible, also called monitoring. Setting values or status requires enabling of the remote control mode (if device is not in „local mode“ or similar). Also see 3.2.

The **4th column** defines a special access condition of an object. The execution of these objects additionally depends on one of the below conditions. If the condition is not given, the object is not executed and the device will return an error message which contains an error code. Meaning of the numbers:

1 = The output/input of the device has to be switched off

(The object is only accepted and executed by the device if the power output/input is in standby mode)

2 = Option „Internal resistance“ has to be unlocked

(The object is only accepted and executed by the device if the option „Internal resistance“ is unlocked)

3 = Transfer of the function data has to be enabled

(The object is only accepted and executed by the device if it has been instructed before by a different object to receive and set function data)

4 = Function manager activated

(The object is only accepted and executed by the device if the function manager has already been activated manually in the device menu or by a different object)

5 = Function manager not activated

(The object is only accepted and executed by the device if the function manager is not active)



Attention!

It is generally required to set the device into remote control mode before sending objects that will change any value on the device.

The **5th column** defines the type of the data in the **data field** of the telegram. Commonly known data types are used.

The **6th column** defines the data length of the **object's data**. For objects with data type „string“ this byte defines the maximum length of the string. The string has to be terminated with an EOL (end of line, =0) or it ends after the given number of bytes. Strings are transmitted in up to three split telegrams if CAN is used. See also „1.9 Message structure for CAN“.

The **7th column** is used to mask out data of type „char“. The mask defines which bits may be set or unset. Also see section „3.3.2 The mask“.

Columns 8 & 9 explain details about the **data field** content.

Some objects use a two-byte time format, which is explained in section „3.4 The time format“.

3.8.2 Object examples and explanations

➤ A description of the object list columns can be found in section 3.8.1.

➤ All numbers are in decimal, if not marked as hexadecimal by a leading 0x.

I. Function manager (objects 56, 58, 73-75, 78, 90-146)

When using the function manager, the order of using the objects becomes very important. Since setup and control of the function manager is complex it is not handled here, but in an external application note (AN001), which you can also find on the included CD in the folder „\manuals\application notes“ or on our web site.

II. Object 54

The primary function of this object is to switch the device to remote control operation or to switch the power input/output on or off. The mask has to be given according which bit is going to be changed.

Example: Activate remote control -> Bit 4 ->Value of bit 4 = 0x10 -> Mask 0x10 -> control byte also 0x10. The object 0x32 will then contain the data 0x1010. Deactivate remote control, the same way: Mask 0x10 -> Control byte 0x00 -> data 0x1000.



Attention!

In order to avoid logical collisions of functions that are assigned to the bits, it is advised to only set 1 bit per telegram!

When reading objects that require a mask, the main mask is also returned, but is only of informative value.

III. Object 56

The bits are here only allowed to be set individually. Else the actions are not executed properly.

IV. Object 73

The time stamp ist only available when using the function manager, else it will be 0. It represents a counter's value of the elapsed time in 2ms steps. Because it is an integer value, the counter will restart at 0 after 65536 x 2ms.

V. Object 77

Reading the alarm buffer will clear it (EL 3000 or PS 8000). Other device series, like PSI 9000, PSI 8000, PSI 800 R and BCI 800 R require initiation to flush the buffer, by setting a bit in object 54. Because it only stores 3 events, i.e. alarms, the first three events are held and any subsequent event will always overwrite the most recent one in position „Last alarm type“.

Example: the object returned 0x0120 in the first two bytes in the data field (last recorded alarm) -> error type 0x01 means, that the error still persists and error code 0x20 says that is is an overtemperature error in the upper power stage of a multi-stage PSI 9000 power supply (see table in section 3.9),.

VI. Objects 39-47

These objects are related to the section 7.6 of the PSI 9000 or PSI 8000 instruction manuals. Events that are triggered by the supervised values will generate an error of type alarm, warning or notification (see PSI 9000 user manual for definition) in the alarm buffer, depending on the configuration with objects 44-46.

The time to give here is a latency for the event trigger. Valid time range: 2ms...100h. For time format see section 3.4.

VII. Objects 21-29

These objects are used to load preset lists into the device, just like you can enter and modify directly at the device. But there is no further control possible, like selecting a preset or switching between presets in order to generate voltage jumps. In order to perform such actions, other objects have to be used.

3.8.3 About user profiles

The device series PSI 8000 and PSI 9000 feature more than one user profile. Up to 4 can be stored and recalled. Every profile contains a set of settings and also a set of preset lists. It means, the user has to be aware which profile is currently selected when writing preset lists into the device or reading them.

The settings and values that belong to a profile, represented by the objects, are colored in the **object lists** (if the list is for a device series that features multiple user profiles).

3.9 Device errors, alarm codes and alarm types

Fehlercode Error code	Kürzel Abbr.	Fehlertext oder -beschreibung / Error text or description
0		Kein Fehler / No error
1 OV		Überspannung am Ausgang (Eingang) / Overvoltage at output (input)
2 OT		Übertemperatur im Gerät / Overtemperature inside the device
3 SYS		Systemfehler / System error
4 U>		Obere Spannungsgrenze überschritten / Upper voltage threshold exceeded
5 U<		Untere Spannungsgrenze unterschritten / Lower voltage threshold exceeded
6 I>		Obere Stromgrenze überschritten / Upper current threshold exceeded
7 I<		Untere Stromgrenze unterschritten / Lower current threshold exceeded
8 SIO2		System Link Mode: Kommunikation gestört / Communication disturbed
9 MS1		System Link Mode: Ein oder mehrere Gerät sind "offline" / One or more units are offline
10 S-OV		System Link Mode: Slave meldet Überspannung / Slave is reporting an overvoltage
11 S-OT		System Link Mode: Slave meldet Übertemperatur / Slave is reporting overtemperature
12 S-PH		System Link Mode: Slave meldet Netzfehler / Slave is reporting mains voltage error
13 S-PD		System Link Mode: Slave ist in Leistungsbegrenzung / Slave reduces max output power
14 S-?		System Link Mode: Slave antwortet nicht / Slave does not answer
17 F01		Interner Fehler / Internal error
19 F03		Interner Fehler / Internal error
20 CAN		CAN: Kommunikation gestört / Communication disturbed
21 FCT		Funktionsmanager: Funktion konnte nicht gesetzt werden / Function manager: function could not be set
22 UDU		Überwachung Sprungantwort: Anstieg U / Step response supervision: U rise
23 UDD		Überwachung Sprungantwort: Abfall U / Step response supervision: U fall
24 IDU		Überwachung Sprungantwort: Anstieg I / Step response supervision: I rise
25 IDD		Überwachung Sprungantwort: Abfall I / Step response supervision: I fall
26 PDU		Überwachung Sprungantwort: Anstieg P / Step response supervision: P rise
27 PDD		Überwachung Sprungantwort: Abfall P / Step response supervision: P fall
28 PH1		Phasenausfall oberes Leistungsteil / Phase loss of upper power stage
29 PH2		Phasenausfall unteres bzw. mittleres Leistungsteil / Phase loss of lower resp. middle power stage
30 PH3		Phasenausfall unteres Leistungsteil / Phase loss of lower power stage
31 OT1		Übertemperatur oberes Leistungsteil / Overtemperature of upper power stage
32 OT2		Übertemperatur unteres bzw. mittleres Leistungsteil / Overtemperature of lower resp. middle power stage
33 OT3		Übertemperatur unteres Leistungsteil / Overtemperature of lower power stage
34 CC		nur UIR Betrieb: Betrieb des Stromreglers nicht erlaubt
35 CP		nur UIR Betrieb: Betrieb des Leistungsreglers nicht erlaubt
36 -		Bat.temp. out of range (Batterietemperatur zu hoch)
37 -		Bat.temp. out of range (Batterietemperatur zu niedrig)
38 -		Bat.voltage out of range (Batteriespannung zu hoch)
39 -		Battery deeply discharged (Batteriespannung zu niedrig bzw. tiefentladen)
40 -		Cell fault in battery (Zellenschluß)
41 -		Temp sensor fault (Temperaturfühler fehlt oder defekt)
42 -		Reverse polarity (Batterie verpolt)
43 -		Battery not connected (keine Batterie angeschlossen)

nur bei Mehrphasengeräten / only at multi-phase models

Gilt für BCI 800 R Serie / Applies only to BCI 800 R series

Table: Alarm codes

How does the alarm management work?

In case of an error according to the adjacent table, it is classified according to an alarm type (see below) and put into an alarm buffer with 3 indexes. This buffer can be read by object 77 (see [object lists](#) for the layout).

It applies, that the first error will be shifted down the buffer by the next error. If the buffer is full, the next error will always overwrite the last error. Reading the buffer will either empty it automatically (with series EL3000/9000 and PS 8000) or it is required to request action by setting a bit in object 54 (with series PSI 9000, PSI 8000, PSI 800 R and BCI 800 R).

What is an alarm type?

About the meaning and the differences of alarms, warnings and notifications at power supplies of the series **PSI 9000**, **PSI 8000**, **PSI 800R** and battery chargers of series **BCI 800 R** please refer to the user guide of your device. Other device types like the **EL 3000 / EL 9000** only use alarms and alarm types 0x01 or 0x02.

Alarm types:

0x01 - Alarm is currently active

0x02 - Alarm is not active anymore

0x10 - Warning currently active

0x20 - Warning not active anymore

0x40 - Notification only

The error type will be returned together with an error code if the alarm buffer is queried from the device and can then be analysed. Warnings and notifications have lower priority than alarms, are particularly overwritten and thus have to be considered as either less important or not important at all.

4. Profibus (IF-PB1)

4.1 General

The Profibus, short for Programmable Field Bus, is an independent field bus standard for a wide range of applications in measurement technique and automation. In compliance to IEC 61158, Profibus guarantees flawless communication between devices of various manufacturers. A lot of different interfaces for PCs, PLCs etc. are available on the market.

The interface IF-PB1 supports Profibus DP (i.e. *decentralised peripherals*), which is especially for fast field bus communication. The data transfer is done via a RS485 connection and operates at bus speeds of up to 12MBaud.

A Profibus is separated into segments. Every segment consists of max. 32 participants. Bus participants are the master, slaves, repeaters and converters. Repeater and converters are required if the number of participants exceeds 32. Every Profibus participant is assigned a dedicated address, resulting in max. 31 participants for the first and last segment and max. 30 participants for middle segments. Addresses 0, 126 and 127 are reserved.

Conclusion: A total of 125 slaves with IF-PB1 cards can be used on one Profibus.

In dependency of the cable length of the bus, baud rates between 9.6kBaud and 12MBaud can be selected for the bus. Notice that a wrong selection of the bus speed, according to the cable length, might not guarantee flawless communication anymore.

The interface card IF-PB1 supports all available bus speeds, it features galvanic isolation between field bus and device of up to 1000V and is supplied with power by the device.

4.2 Technical specifications

Ordering code	IF-PB1
Communication profiles	Profibus DP-V0, Profibus DP-V1 Class 1
Interface type	9 pole D-SUB socket
Communication hardware	RS485
Network topology	Line (without repeater), Tree/Line (with repeater)
Number of participants	max. 31 (without repeater), max. 125 (with repeaters)
Participant type	Slave
PNO Identification number	A000 (hex)

4.3 Supported devices

IF-PB1 supports and is supported by following device series (date 08/2012):

- PSI 9000
- PSI 8000
- PS 8000

4.4 Master-Slave communication

4.4.1 Cyclic

The Profibus-DP distinguishes between master and slave participants. The master controls the bus communication and requests data from or sends data to the slaves. In a typical master-slave system, input data, output data and diagnosis data are exchanged between the master and all his assigned slaves. The master, for example a PLC, stores input data in its internal memory for the control application. Output data are transferred to the slaves with the next transfer cycle.

So the master always contains a full set of data from its slaves in its memory, but these data have a delay of one cycle.

In case of the IF-PB1, the actual values and the device condition are constantly, i.e. cyclically, transferred to the master.

4.4.2 Acyclic

Alternatively to the cyclic data exchange, data can also be transferred acyclically. This has the advantage to access certain single parameters or set values of the field bus device. The acyclic data exchange can reduce the amount of transferred data to a minimum and thus decrease bus load significantly.

4.5 Cabling / Termination

The connection of the IF-PB1 to other slaves on the bus or the master system is done with a 2-wire, screened cable. Two variants, type A and type B, of the bus cable are specified in the IEC 61158. Cable type B is not recommended and should not be used for new installations.

Cable type A:

- Cable: twisted pair, screened, 1x2
- Wave resistance R_w : 135-165 Ω at 3..20MHz
- Capacity: < 30pF/m
- Loop resistance R : 110 Ω /km
- Wire cross section: >0.34mm²

The D-Sub socket on the IF-PB1 connects the device with the Profibus cable.

It is used to connect the ingoing bus wires as well as the outgoing ones. The wires „Ingoing“ and „Outgoing“ must not be exchanged, because a standard Profibus plug cuts off the outgoing wires as soon as the bus termination is activated at a certain participant. The bus is not interrupted when disconnecting one device (power-off or disconnection). This applies when multiple units with IF-PB1 card are connected to the bus.

Activating the bus termination resistor will prevent reflexions on the bus ends and provides a clean idle level on the bus.

4.6 Transfer speed and delays

A list of available transfer speeds for the IF-PB1 Profibus card can be found in section 4.6.1. Refer to your device operating guide resp. the interface cards manual for details on how to access and alter the communication settings.

The **transfer speed** only defines how fast data is transported between the Profibus master and the Profibus slave (i.e. gateway). The Profibus gateway chip processes the data in the Profibus transmission and forwards them to the device with a fixed baudrate of 57600. The same applies for the reverse direction. The data on the Profibus is refreshed by the gateway within a defined period of time, but the data itself is depending on how often it is refreshed by the device. That interval varies from series to series.

The slave automatically puts data (status, actual values) on the bus in a certain interval (DP-V0), in this case 1x per cycle. That data which is also read once per cycle from the device is buffered in the gateway. One cycle is about 660ms.

Furthermore, if data is sent to the device by DP-V1 (set values etc.), these are also forwarded to the device once per cycle.

This results in certain **delays**, until a set value „arrives“ on the DC output resp. DC input of the devices. The delay comes from the data transmission time to the gateway, processing time in the gateway, transmission time to the device and processing time in the device. If, for example, a set value would be sent, then the actual value which belongs to the most recent set value can not be read within the same cycle. Why? Because in the first cycle, the set value is received from the master and sent to the device. In the next cycle, the actual value is measured. That measured value may still not belong to the new set value, so it might only be read in the next cycle. So in the worst case it may take about three cycles to get the corresponding actual value. It is similar to setting a status like „output off“ and reading the device status with DP-V0 to verify that the output has been set to off.

In the standard configuration of the IF-PB1, an operation like „set a value and then wait for the corresponding actual value to appear on DP-V0 data“ usually takes 1.2s...1.5s. This can only be decreased with optimized scripts, of which one can reduce that time down to approx. 0.35s. The optimized scripts will still deliver a new actual value with every cycle, but the status and/or alarm condition is only updated every 2nd or 3rd cycle.

The optimized scripts can be uploaded to the gateway chip by the user herself/himself and are available upon request.

4.6.1 Influence of cable lengths

For the data transmission speed and thus time between Profibus master and slave, the length of the cable determines the max. speed that can be used without problems. The table below lists common Profibus speeds of which some are available for the IF-PB1:

Transfer speed (kBit/s)	max. cable length (m)
9.6	1200
19.2	1200
45.45	1200
93.75	1200
187.5	1000
500	400
1500	200
3000	100
6000	100
12000	100

4.7 Operation modes

A jumper matrix on the IF-PB1 board is used to switch between different operation modes of the interface card. Those modes are for normal operation or maintenance purposes.

The **RS 485/D-Sub port** is used for:

- Profibus (field bus communication)

The **USB port** (virtual serial port) is used for:

- Script updates with WINGATE software or Script Programming Tool (SPT)
- Firmware updates of the device controller (Update Tool)
- Firmware updates of the Profibus stack with Firmware Download Tool (FDT)

The card features a LED, which indicates a proper bus connection and communication readiness by being not lit. It is lit during the initialisation sequence of the Profibus slave controller or when bus connection errors occur. Thus it will be lit for a certain time after the device is switched on.

As soon as the slave units are disconnected from the bus, it will also be lit indicating the error. A detailed description of the momentary condition of the communication is commonly provided by the software configurators of the master systems.

4.8 Profibus address

Every bus participant is required to have a unique bus address. This address has to be defined before any unit is connected to the bus or at least before the unit is running on the bus. It is advised to set the Profibus address (not to mix up with the „device node“) during the installation of the unit, as described in the user guide of the device. No address must be double. We recommend to assign addresses in ascending order, whereas address 0 is reserved for a PC or programming device.

The assignment of all addresses for the Profibus participants has to correspond to the address assignment in a project on the master system. Only then a secure identification of the power supplies is provided.

Note: The address that is selected in the setup menu of the particular power supply is stored and will be restored after the next start or after a reset by the reset button.

4.9 Communication with the master

Note

The data, as transfered in the datagrams which are described below, is of different type. Set values and actual are, for example, of type integer and contain a per cent value. Refer to complete sections 1.7 and 3.3 - 3.8 for more information.

4.9.1 Cyclic field bus data

Cyclic field bus data, which are transferred from the slave (IF-PB1) to the master, are actual values and the device conditions, in this case. But this will only happen, if the IF-PB1 accepts the configuration that is sent by the master. The type of the data for the cyclic data exchange is sent by the master in a configuration tele-gram. The data width of the particular module can be obtained from an object list which is related to the de-vice series. The width can also be obtained by help of the hardware configurator from the properties of a particular module slot. It corresponds to the address width of, for example, a PLC.

Rules given in the object list also apply for data formats and the actual values resolution of the power supply. These have to be considered when programming user applications.

4.9.2 Acyclic field bus data

Acyclic data exchange between master and slave only happens upon request of the master. The IF-PB1 supports the communication to a class 1 master, as well as to a class 2 master. The datagrams listed in section 4.9.2.1 are suitable for this type of data exchange and are separated into several functional groups (also see the device related object list).

4.9.2.1 Datagram groups

Device information DP-V1 (outputs from slave)

Datagram	Slot	Index	Length	Type	Object in list *	Access from host
Device type	3	0	16	String	0	Read
Device serial number	3	1	16	String	1	Read
Device nom. voltage	3	2	4	Float	2	Read
Device nom. current	3	3	4	Float	3	Read
Device nom. power	3	13	4	Float	4	Read
Device article number	3	4	16	String	6	Read
Device firmware version	3	5	16	String	9	Read
Momentary set values	3	9	6	Integer	72	Read
Device alerts	3	10	6	Byte	77	Read

Control DP-V1 (inputs to slave)

Datagram	Slot	Index	Length	Type	Object in list *	Access from host
Set device condition	3	6	2	Byte	54	Write
Set value of voltage	3	7	2	Integer	50	Write
Set value of current	3	8	2	Integer	51	Write
Set value of power	3	15	2	Integer	52	Write

* See external [object lists](#) (PDF), also see section 3.8

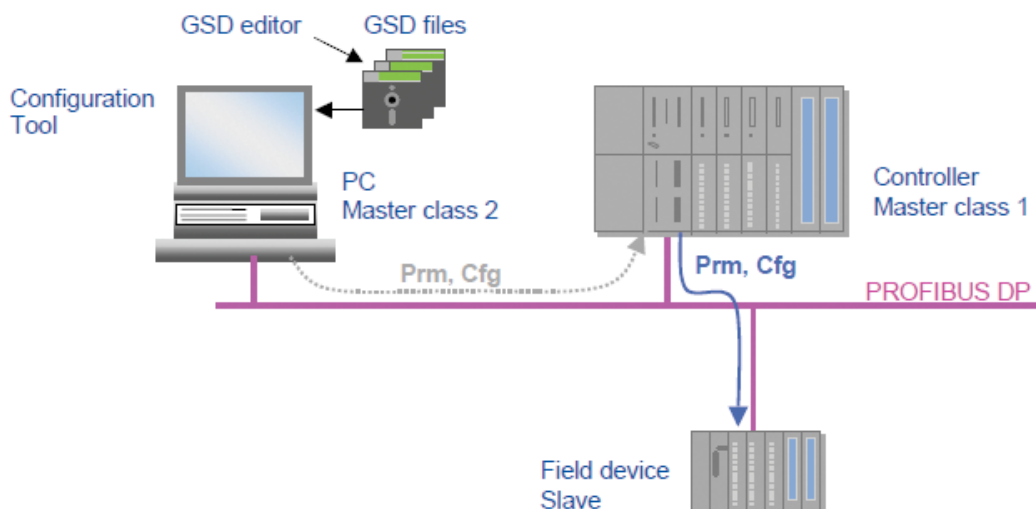


Figure 1: Profibus scheme

Alerts and actual values DP-V0 (outputs from slave)

Datagram	Slot	Index	Length	Type	Object in list *	Access from host
Actual values (U,I,P)	2	-	6	Integer	71	Read
Device condition	2	-	6	Byte	70	Read

* See external [object lists](#) (PDF), also see section 3.8

4.9.3 Using the datagrams

If the user wants to read or write a datagram acyclically in the user application, means with DP-V1 Master Class 1, the particular datagram has to be selected by its INDEX (see datagram groups above), the SLOT and access type (READ or WRITE). When reading, the length of the data to read is required as auxiliary information. The data length is an input parameter of the standard SFBs (system function block) of PLC software.

The data to write will usually be hexadecimal bytes, also in the examples below. They can be converted to decimal values, if only decimal input is available. The contents of the data is defined in the so-called [object lists](#), on the basis of the communication protocol that is used with other interface cards. The Profibus message to and from the slave just contains the data part. See sections 3.2 and 3.3.2 for important details.

4.9.3.1 Examples

1) Activate/deactivate remote control

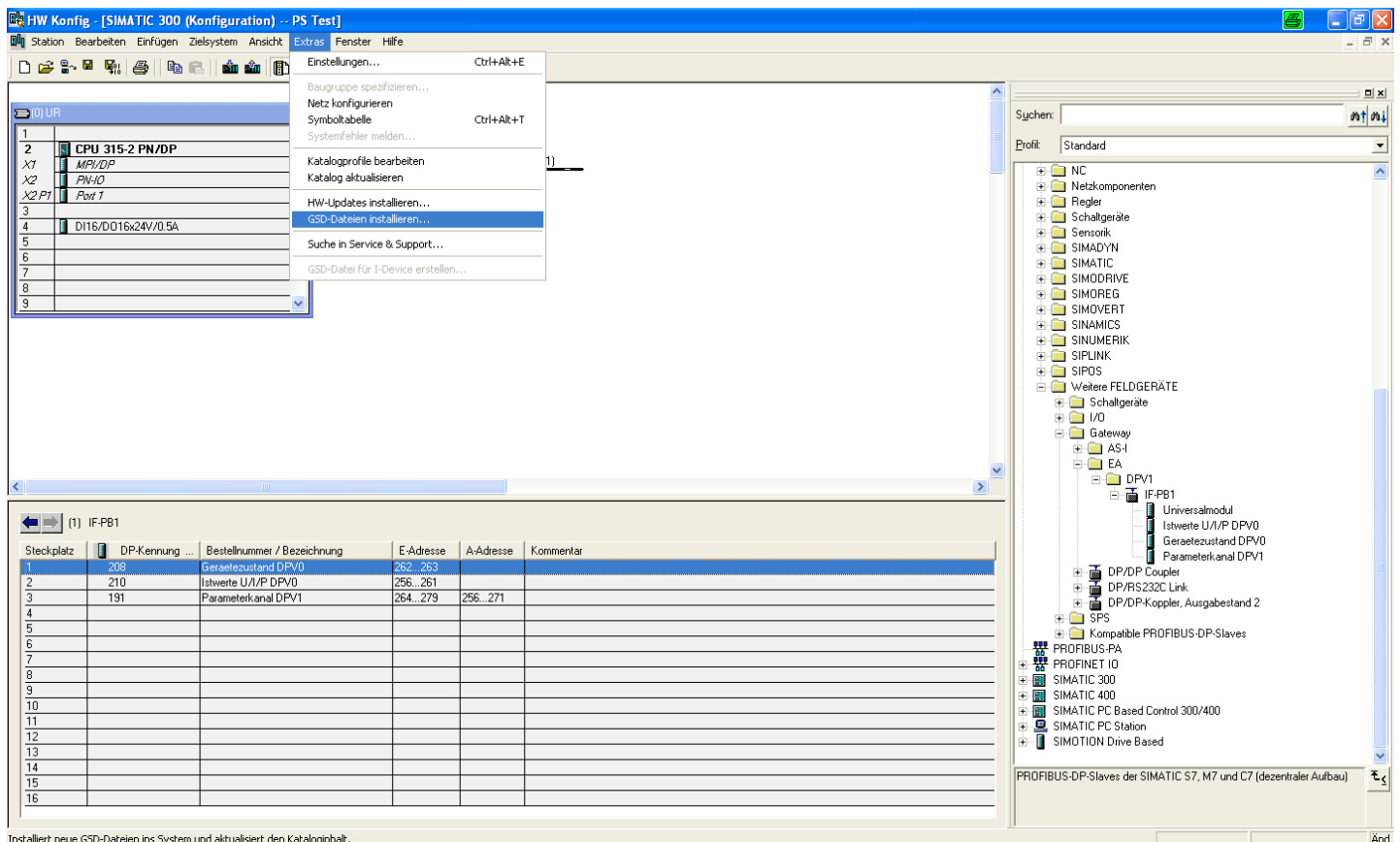
Remote control condition is absolutely required if you want to control the device. So the condition of the device is changed and „Set device condition“ has to be used. This is related to object 54 in the object list and has a length of 2. According to object 54 and in order to switch remote control on or off, the mask byte has to be 0x10 and the control byte 0x10. So the bytes 0x10 0x10 are sent to slot 3, index 6. To deactivate remote control, bytes 0x10 0x00 would then be sent to slot 3, index 6.

2) Switch DC output on or off

Switching the DC output on or off changes the device condition, hence datagram „Set device condition“ is the choice. It is related to object 54 and has a length of 2. According to object 54 and in order to switch the output on the mask byte has to be 0x01 and the control byte 0x01. These bytes are sent to slot 3, index 6. Setting the output off would require to send the bytes 0x01 0x00 to slot 3, index 6.

3) Set voltage to 40V

This requires to translate a real voltage value to a per cent value first. In case you have a 80V model, then 40V would be 50%. In case of a 720V model, 40V would be 5.55%. See section „1.7 Translating set values & actual values“ for details about set value translation. With 50% the hex value would be 0x3200, sent as bytes 0x32 0x00 to slot 3, index 7.



Installiert neue GSD-Dateien ins System und aktualisiert den Kataloginhalt.

Figure 2: GSE installation

4.10 Example project planning with a Siemens PLC

4.10.1 Integrating the power supply into a Profibus

In order to integrate a power supply or similar by means of an IF-PB1, a hardware configurator is required, which is part of the software package belonging to the Siemens PLC. Other configurators are available from companies like Beckhoff or Hilscher. Besides this, a GSD (Generic Station Device) file is required which is delivered on the CD included with the IF-PB1 or can be obtained upon request as well as from the website of the PSU manufacturer. The file is named PBPSA000.GSD (german version) or PBPSA000.GSE (english version).

This file is used to announce the communication modules and required parameters of the power supply to the user application in the configurator. The cyclic data exchange on the bus will be defined.

4.10.2 GSD file installation

The GSD file, contained on the CD that is included with the IF-PB1, is installed in the master application with the menu command „Extras -> Install GSD files“. See the following screenshots from the german version on next page.

After the successful installation of a GSD file, the Profibus interface of the power supply can be found in the hardware catalogue in „Profibus-DP -> Other field devices -> Gateway ->EA ->IF-PB1“.

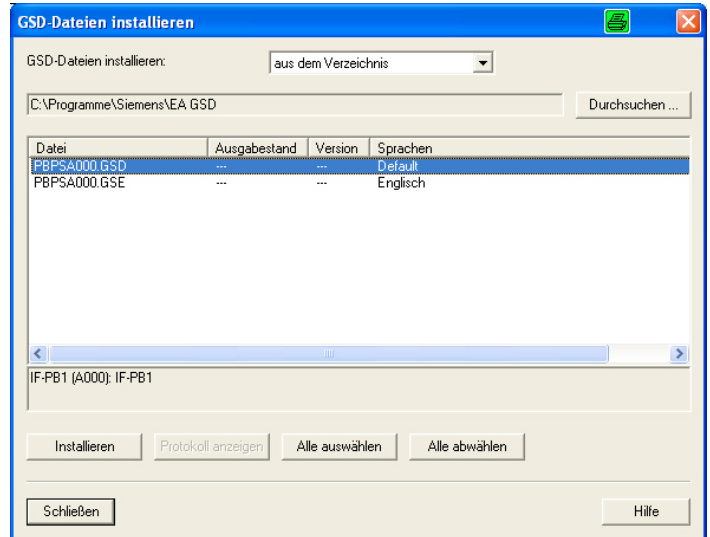


Figure 3: Choose GSE install path

4.10.3 Module placement

The modules, which are now located in the above (section 2.2) selected path, need to be announced to the data exchange control, for example by drag & drop in the module placement list. In order to enable full communication with the device, the modules have to be placed as follows:

- Slot 1 : Actual values U/I/P DPV0
- Slot 2 : Device condition DPV0
- Slot 3 : Parameter channel DPV1

Also see figure 4 below.

The sequence of the module placement can vary. It means, the modules „Device condition“ and „Actual values“ can be exchanged, while the module „Parameter channel“ must be placed in slot 3.

If module „Parameter channel“ is not placed, the device can not be controlled. With this, acyclic data exchange is disabled and the red LED on the interface panel will be lit.

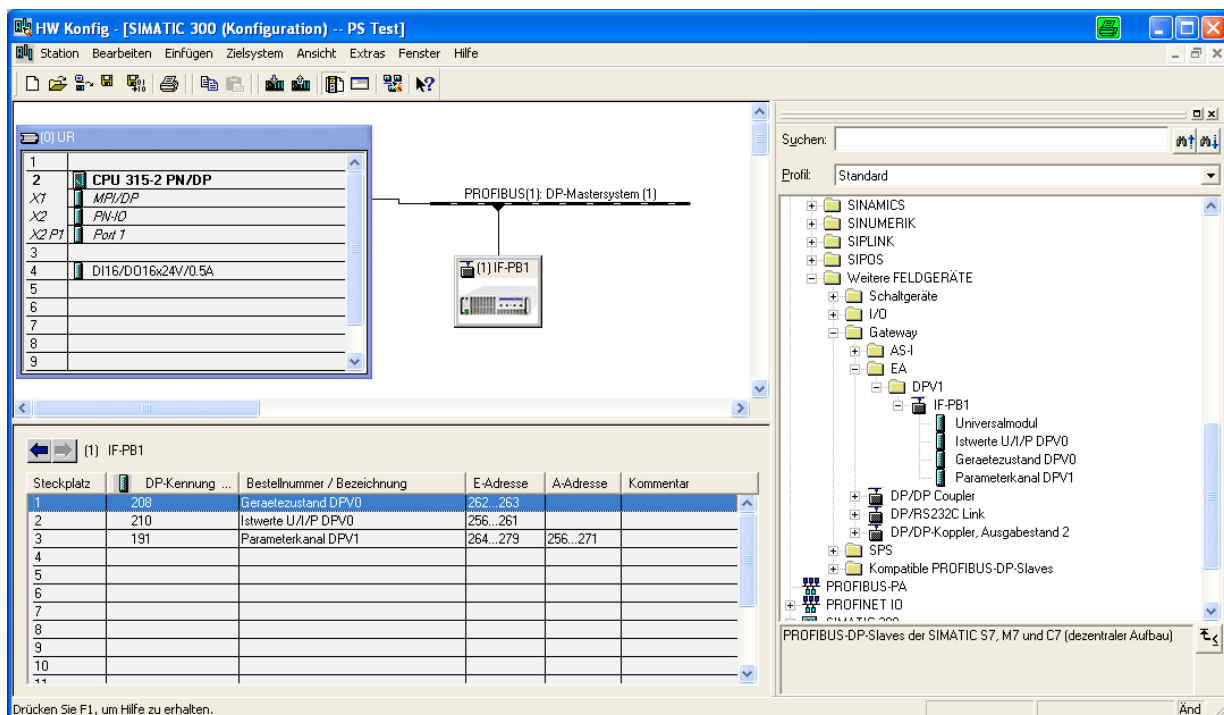


Figure 4: One possible module placement

4.10.4 Addressing the cyclic modules

For the modules of cyclic data exchange it applies generally:

- All modules in the hardware catalogue are assigned to a logical address
- The logical addresses build a part of the control peripherals
- The peripheral addresses of the single modules may not collide and thus not assigned twice
- The assigned peripheral address has to be used in the user application to access data

4.10.5 Addressing the acyclic module

For the module of acyclic data exchange it applies:

- The peripheral address(es) must not conflict with the ones of cyclic modules
- Because this module is a mixed type (input/output), the addresses of input and outputs must also not conflict. This is based upon the design of the Siemens communication blocks (SFB52 and SFB53) for acyclic data exchange
- When accessing data access via an interface which is standardised according to PNO AK 1131, an ID will be requested first which consists of the lower one of both addresses. The address is the basis and the first part of two information, that are required for acyclic data exchange
- The second information is the datagram number that shall be read or written. The standard here speaks of a so-called INDEX

4.11 Diagnosis

4.11.1 Standard diagnosis

The standard diagnosis is a telegram of 6 bytes. The telegram is exchanged during the initialisation of the IF-PB1 between the interface and the master, unnoticed by the user. But it can be interpreted by an adequate engineering tool. In STEP7 it is done with „Target system -> Show present participants“ and can be displayed by selecting the particular field bus device with „Target system-> Diagnosis/Setup -> Component condition“ (see fig. below). The diagnosis telegram is furthermore a part of the permanent data exchange that happens in case of an error where the IF-PB1 will send additional information over the field bus. See figure 5 below.

For the IF-PB1 the diagnosis telegram will look like this during normal operation: 0x80 | 0x0C | 0x00 | 0x02 | 0x0A | 0x00

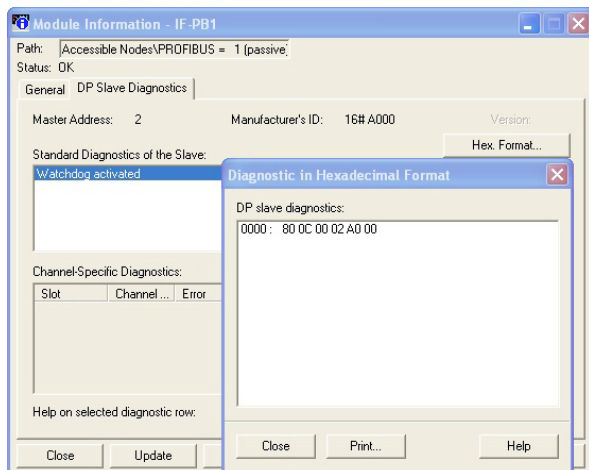


Figure 5: Standard diagnosis

4.11.2 DP-V1 alarm management (extended diagnosis)

In excess of the standard diagnosis, as specified in the communication mode DPV0, the user can receive extended diagnosis information from the IF-PB1.

In communication mode DPV1 the so-called device based diagnosis is build from alarm blocks and/or condition reports. To activate this syntax the bit DPV1_Enable in the parameter telegram has to be set to 1.

The IF-PB1 makes use of the condition report. It is defined in the IEC61158 as follows:

Byte	Bit	
1	0	Station does not exist (set by master)
	1	Slave is not ready for data exchange
	2	Configuration data not concurring
	3	Slave has extended diagnosis data
	4	Requested function not supported by slave
	5	Invalid response from slave (set by master)
	6	Wrong parameterisation
	7	Slave is already parameterised by a different master (set by master)
2	0	Slave has to be parameterised again
	1	Static diagnosis
	2	Set to "1"
	3	Watchdog active
	4	Received Freeze command
	5	Received Sync command
	6	Reserved
	7	Slave is deactivated (set by master)
3	7	Diagnosis overflow - Slave has more diagnosis information than fits the telegram
4	0...7	Master address after parameterisation (0xFF without parameterisation)
5	0...7	Ident number high-byte
6	0...7	Ident number low-byte
7		Header: - The header shows the block length of the extended diagnosis, including the header byte - For this module this value is 0x05 (bytes 7...11 = 5 bytes)
8		Status_Type: The value is fixed to „0x81“ with following bit significance: - Bit 7 = 1: „Condition“ - Bit 0 = 1: „Condition report“
9		Slot_Number: Value is „0x00“
10		Specifier - A new error will be marked in the specifier with „0x00“ (Condition coming) - A removed error will be marked in the specifier with „0x02“ (Condition leaving) - If no error is reported, the specifier value will be „0x00“
11		User byte: Detailed explained below

Continued on next page...

The extended diagnosis for an IF-PB1 will result like this, according to the table above:

0x05 | 0x81 | 0x00 | 0x00 | **0x??**

The last byte defines the actual error information in form of a code, which either represents a communication error (see table in 3.7.2 on page Seite 32) or a device error (see table in 3.9).

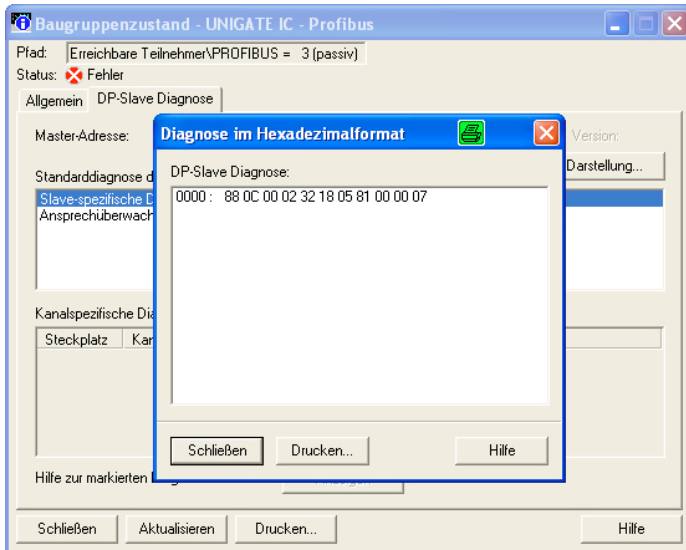


Figure 6: Extended diagnosis (german screenshot)



Elektro-Automatik

EA-Elektro-Automatik GmbH & Co. KG

Development - Production - Sales

研发 - 生产 - 销售一体化

Helmholtzstraße 31-33

41747 Viersen

Germany

Phone: 02162 / 37 85-0

Fax: 02162 / 16 230

ea1974@elektroautomatik.de

www.elektroautomatik.de