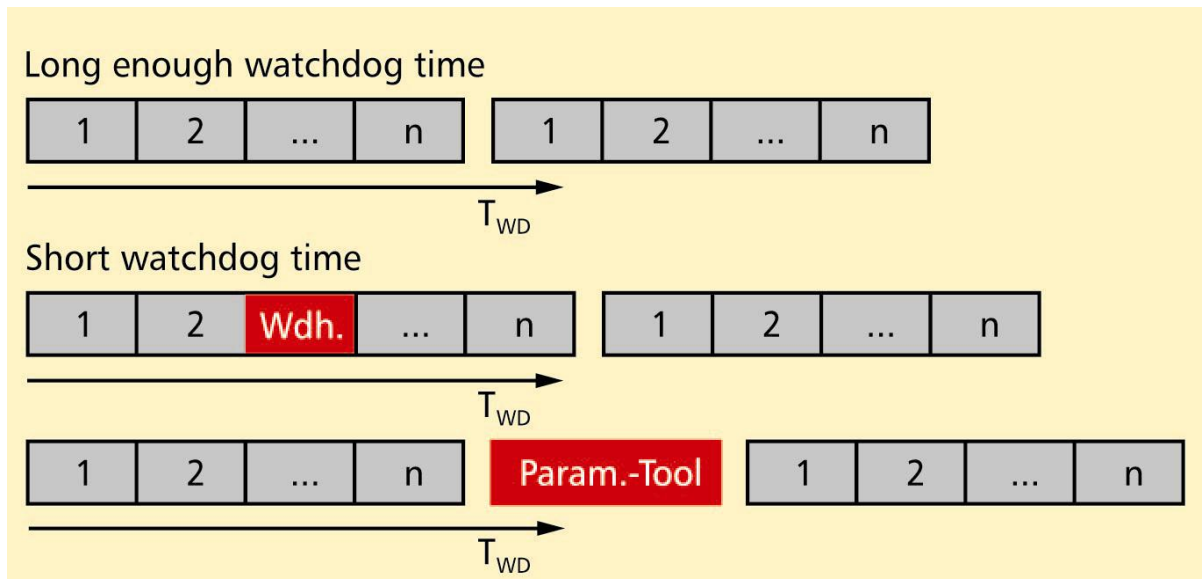


Bus systems are easy to use since communication between devices takes place completely automatically. Fine-tuned mechanisms for preventing and correcting errors ensure reliable data transmission. Tools for automatically configuring the data exchange process make it unnecessary for you to have detailed knowledge of the protocol, and they help save time and money. Nonetheless, communication problems do occasionally occur in some networks. Their causes are often simple, but they can't always be corrected using on-board means. Since so much tool support is provided during the planning phase, there is often a lack of understanding regarding the fundamental relationships within a Profibus network. The second part of the troubleshooting guideline from our March issue can help you here.



Figur 1: Profibus communication cycle: The first line shows the standard flow of communication; the lines underneath illustrate the effects of delays due to repeat telegrams or additional stations.

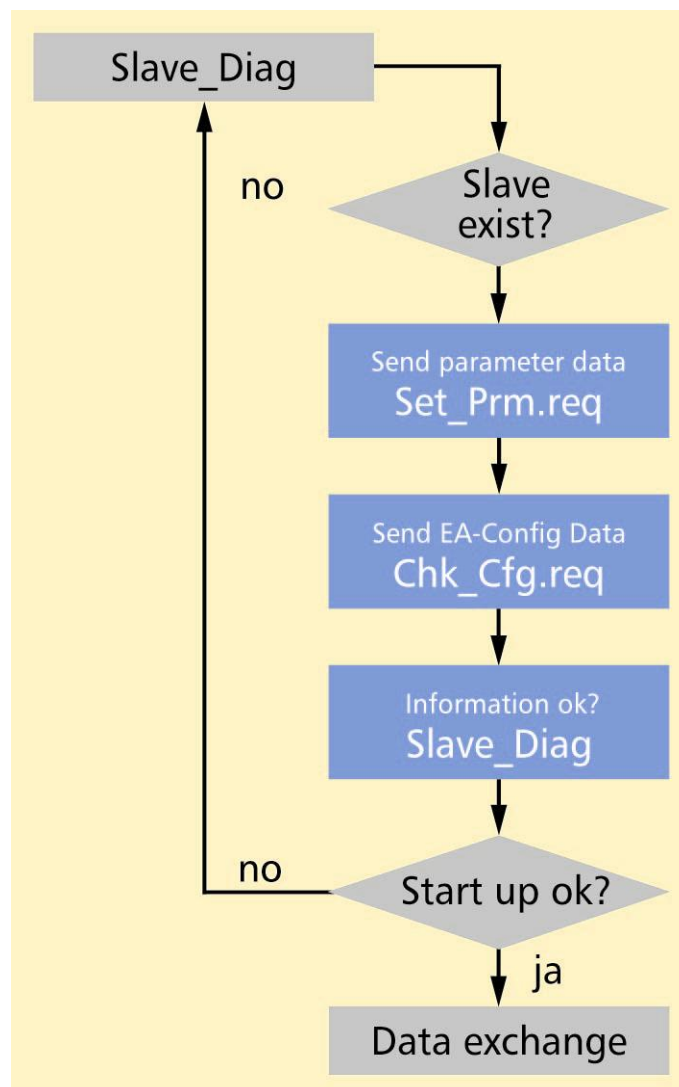
Disruptions must be corrected quickly to keep the costs of production downtimes in check. It is tempting to try to correct problems by means of trial and error. But while indiscriminately changing settings may improve the situation in the short term, it can also cause the Profibus error handling mechanisms to kick in at the wrong time, if at all. If this happens, production will halt again after a short time. With some basic knowledge and the appropriate tools, you can avoid having to exchange devices unnecessarily. Used in advance, such knowledge and tools guarantee stable communication for any application and allow you to eliminate the causes of potential disruptions.

### Where are the disruptions coming from?

The sources of communication problems can be roughly divided into two categories: disruptions due to environmental factors and disruptions due to unsuitable

communication parameters. The first category was discussed in part 1 of the guideline (eA 3/2004, p. 60); the second category will be addressed in the following. Depending on when they occur, communication problems may be attributed to different types of disruptions:

- Case 1: Communication with individual stations is permanently disrupted. This indicates a network planning error.
- Case 2: The disruption occurs irregularly, communication occasionally fails, it is not possible to assign the problem to specific stations. In this case, you should investigate the timing situation in the network. It is important to determine whether the number of stations on the bus has remained the same, or whether the disruption is associated with the addition or removal of a station. In any case, your investigation should start with a bus traffic log which includes time stamps. A tool like the mobile Softing Analyzer can create such a log very easily. The analyzer can be non-reactively connected to the system via a Profiprobe cable. Without having its own station address, the analyzer listens to the traffic on all stations and stores the recorded messages together with exact time stamps in bit times.



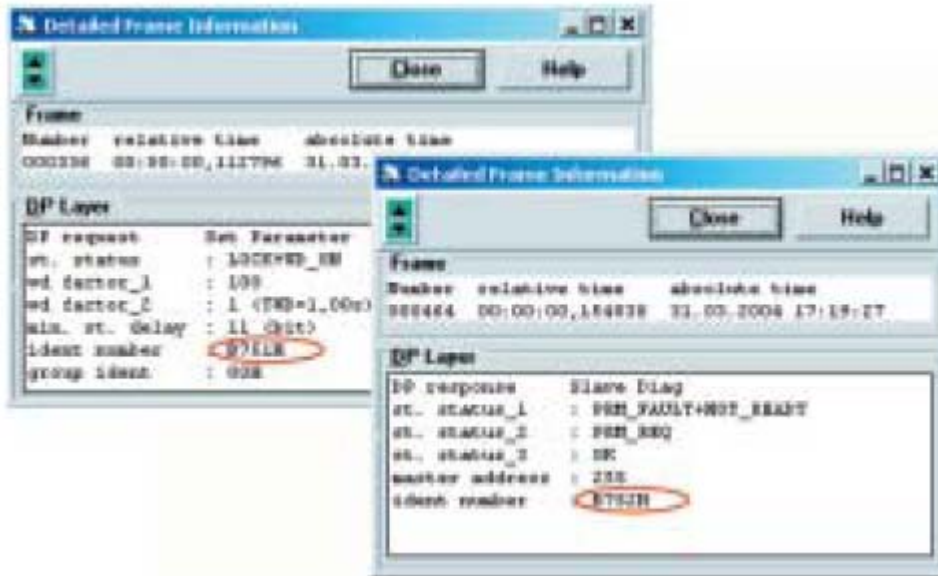
Figur 2: Starting Profibus communication with a slave: If the network planning information is correct, the slave goes into data exchange mode; otherwise, it remains in start-up mode.

### *Practice plus*

- Static communication problems in Profibus networks can be easily investigated with the mobile Softing Analyzer, which can be connected non-reactively via a Profiprobe cable. Without having its own station address, it will listen to the traffic on all stations and create a log with exact time stamps in bit times.
- Sporadic communication problems may be solved with an appropriate "watchdog time." Experience has shown that the watchdog time should be at least 125% of the data cycle time. In networks where additional masters occasionally act as parameterization devices, the watchdog time should be longer than the target rotation time.

### *Parameterization and configuration errors*

Communication between masters and slaves in a DP network is cyclic, meaning that the master sends output data for each slave and the slave immediately responds with input data. Once the last station has been attended to, a new cycle begins (Figur 1, first line). In order to guarantee the rapid exchange of data, only the most critical checks are carried out during transmission. The slave type and connected modules are checked prior to the data exchange during the start-up phase. All critical messages are contained in diagnostic, parameterization and configuration telegrams (Figur 2). The diagnostic telegram initially checks whether a DP slave is connected to the bus at the address defined in the master. If there is no response, the control system receives a diagnostic message of "station non-existent." This may be because the slave's station address has been incorrectly set (on the device itself or in the control system). The bus monitor's "live list" shows which slave stations are present on the bus. Application-oriented station names make it possible to quickly compare the target and actual structure of the bus. If the slave can be reached but data cannot be exchanged, the diagnostic telegram provides further information on the possible causes of the error. Error codes such as "Prm\_Fault" or "Cfg\_Fault" indicate discrepancies in parameterization or configuration. In the parameterization telegram, the device type is queried using the "Ident\_Number." If the code in the parameterization telegram sent by the master deviates from the one stored in the slave, communication will not be possible because the planned slave type is not identical to the slave that is actually installed. The diagnostic telegram contains the number stored in the slave. Thanks to the analyzer's plaintext decoding, immediate comparisons can be made without the need for additional tools (Figur 3). You can therefore solve the problem directly by modifying the network plan or exchanging the field device. The "Cfg\_Fault" message in the diagnostic telegram indicates a similar problem: When the cycle starts, the master also checks whether its network plan is identical to that of the installed slave. The basis of this is an encoded module display which is stored in the control system. When the slave is activated, it carries out a self-test to detect the modules which are actually connected. The two codes are compared during booting. If the planned and installed modules are not the same, the exchange of data is prevented with the diagnostic code "Cfg\_Fault" (configuration fault), because the discrepancies may cause a malfunction. This problem can be solved by modifying the network plan or changing the installed device.



Figur 3: The bus analyzer can be used to compare planned and installed slaves based on their "Ident\_Number" without needing to access information from the planning software.

### Timing is everything – Avoiding sporadic errors

Besides the static communication problems described above, difficulties frequently arise which have only a short-term effect, but which are enough to throw the control system off its stride. These problems can cause the data exchange with the slaves to suddenly cease for no apparent reason and then start up again. Depending on its configuration, the control system may go into error mode in this case, even if the bus has been error-free again for a while. After the error has been acknowledged, everything functions normally again. The response monitoring system in the DP slave plays an important role here. With the "watchdog timer," the slave monitors whether it is addressed at least once by the master within a certain period of time ("watchdog time", Twd) (Figur 1). If this is not the case, the slave's outputs go into safe mode and its bus interface is reset. In order for communication to continue, the slave must be restarted. The timer value also determines the shut-down time of the slave outputs if the master fails. This is why small values are preferable from a safety point of view. If the value is too small, however, communication will not be possible. In this case, the slaves will not get past the start-up phase and will continue to reset themselves. If the value is not high enough, even small irregularities in the communication process will be enough for the timer to respond.

### What small value is large enough?

In order for data to be exchanged, the watchdog time must at least cover the interval between two data telegrams to the same slave. However, any delay in the data exchange – because of repeat telegrams for error correction, for example – can cause the watchdog time to be exceeded. Additional time must therefore be allotted for error correction. The greater this additional time is, the more stable the communication will be, but the more sluggish the reaction will be in the event of an error (Figur 4). Practical recommendations call for at least 125% of the data cycle time. The analyzer makes it easy to determine whether the watchdog timer setting is too low, which could cause communication problems. It is important for the telegrams to have exact, reliable time stamps so that the respective intervals can be precisely established. The plaintext display of the parameterization telegram will tell you the watchdog time. The interval between two telegrams sent to the same slave can be



transmission media. This results in the repeated failure of sections of an installation, which always leads to production downtimes. However, it is easy to determine the operation values of a network. Many problems can be avoided if you pay heed to the simple relationships between a few parameters. Experience has shown, however, that the necessary knowledge is not yet sufficiently widespread. According to the Profibus specialists at Softing, the technical acceptance of a network after installation is a reliable way of avoiding this dilemma. Troubleshooting training for maintenance teams, like that provided by Softing, offers a more sustainable solution to the problem. These training courses give participants compact basic knowledge of the relevant Profibus structures and teach them how to quickly solve problems using the appropriate tools.